Administrative Tasks

FileServ Version 4.3
February, 2001
600716 Rev A

Trademark Notice

AMASS, DataMgr, EMASS, FileServ, and VolServ are either trademarks or registered trademarks of ADIC, Advanced Digital Information Corporation. DAS is a trademark of Grau, an ADIC subsidiary. All other product names and identifications are trademarks or registered trademarks of their respective manufacturers.

Copyright Notice

ADIC
10949 East Peakview Ave.
Englewood, CO 80111 USA
Phone: 303-792-9700
FAX: 303-792-2465

U.S. Government Rights Restricted

Use, duplication, or disclosure of either the software or documentation is subject to restrictions set forth by the U.S. Government in FAR 52.227-19(c)(2) and subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 52.227-7013 and/or in similar or following clauses in the FAR, DoD, or NASA FAR Supplement.

Technical Assistance

ADIC Technical Assistance Center:

- In the USA and Canada, call 1-800-827-3822.
- Outside the USA and Canada, call 303-874-0188 or toll-free 00800-9999-3822.
- Send e-mail to: techsup@adic.com.

Documentation

Although the material contained herein has been carefully reviewed, ADIC does not warrant it to be free of errors or omissions. We reserve the right to make corrections, updates, revisions, or changes to the information contained herein.

- Send e-mail to: swpubs@adic.com

READER COMMENT FORM

ADIC includes this Form in an effort to provide the best possible documentation to our customers. Please take a few moments to mail or FAX your response to:

ADIC
Software Documentation
10949 East Peakview Ave.
Englewood, CO 80111
FAX: 303-792-2465
E-mail: swpubs@adic.com

Question	Circle One	
Information was complete.	Agree	Disagree
Information was easy to find.	Agree	Disagree
Information was easy to follow.	Agree	Disagree

Is there anything you especially *like* or *dislike* about the organization, presentation, or writing in this manual?_____

_____

_____

_____

_____

Book Title	Document Number
Customer Name	Telephone
E-mail Address	
Company Name	
Address	
City, State, Zip	

NOTES

Contents

Preface

System Tasks 1

Operating Tasks 2

Utilities 3

Trouble-shooting Tools 4

Index

Preface

NOTES

Purpose of This Book

This book describes the management tasks, utilities, and troubleshooting tools used in operating FileServ.

Who Should Read This Book

This book is written for the system administrator who is operating and troubleshooting FileServ.

It assumes the administrator has a strong familiarity with:

- UNIX operating systems.
- Applications running in their site environment.

How This Book is Organized

This book contains the following chapters:

Chapter 1: Initial Tasks — Initialize FileServ, configure interface with VolServ, perform start up and shut down, configure system parameters, and tune system for better performance.

Chapter 2: Operating Tasks — Manage: DataClass Groups, files, migration, data, and media.

Chapter 3: Utilities — Using the FileServ utilities.

Chapter 4: Troubleshooting Tools — Troubleshoot operating problems.

Conventions

The conventions used throughout the FileServ technical books are listed below:

Convention	Example
Screen text, file names, program names, and commands are in `Courier` font.	`FS0000 03 204773 fsmedinfo completed: Command Successful.`
The root prompt is shown as a percent sign.	% **su root**
What you should type in is shown in `Courier` **bold** font.	**fsadmret**
Site-specific variables are in a *Times italics* font.	**fsaudit** ***newfilesystem***
A backward slash (\ ) denotes the input is continued onto the next line; the printed page is just not wide enough to accommodate the line.	% **remsh nodename -n dd if=/dev** \ /***tapedevicename***/**bs=20b | tar xvfb \ - 20** (You should type the entire command without the backward slash.)
Pressing <Return> after each command is assumed.	
A menu name with an arrow refers to a sequence of menus.	Edit Policy -> Add Library

Books

The books described below are part of the technical documentation set, and are shipped on CD:

FileServ Overview
An overview of FileServ. Contains a glossary.

Installing FileServ
Describes: server requirements, installation procedures, configuration parameters, and tools to troubleshoot install problems.

Error Messages
Summarizes error messages.

Administrative Tasks
System tasks, operating tasks, FileServ utilities, and troubleshooting problems.

Command Reference
An alphabetical list of FileServ commands.

quick reference card
Summarizes commands.

Online Books

The documentation CD contains FileServ book files and Adobe® Acrobat® Reader. The Reader allows you to view and navigate the online documentation files yet preserves the page design and graphics from the printed books.

Related Publications

The publications described in the table below are created and distributed on an as-needed basis.

Related Publications	Description
"Release Notes"	For each version of FileServ, the "Release Notes" contain: • Summary of enhancements. • Describes: - Fixed problems. - Known problems. - Installation and configuration issues. • Lists: - Operating system patches. - System requirements.
"Product Alerts"	Informs customers of technical problems and solutions.
"Product Bulletins"	Conveys technical information — not problems — to customers.

Contact Publications Department

To make corrections or to comment on FileServ publications, please contact Software Technical Publications at our email address: swpubs@adic.com.

Secured Web Site

To receive access to the secured site on our home page containing technical product information, send email to swpubs@adic.com and include your: name, company, address, telephone number, fax number, FileServ serial number (or enter "reseller" if you are not a customer), and your email address. In return, we will send you instructions and a password.

1

System Tasks

Roadmap

Topic	Refer To Chapter
Initialize FileServ, configure interface with VolServ, perform start up and shut down, modify system configuration, and tune system for better performance.	1
Manage: DataClass Groups, files, migration, data, and media.	2
Using the FileServ utilities.	3
Troubleshoot operating problems.	4

Note

For complete information on the FileServ commands used in these tasks, refer to the *Command Reference* book. Status for the commands used in these tasks is returned to the command line.

Note

For complete information on the menus used in these tasks, refer to the *Using the FileServ GUI* book.

Initialize FileServ

Running FileServ for the first time involves the following tasks listed in the table below:

Access the Host Server

To access the host where FileServ has been installed to transfer files *to* or *from* the migration directories, use one of the following methods:

- UNIX Remote Copy Protocol (`rcp`)
- File Transfer Protocol (`FTP`) session with put/get
- UNIX copy (`cp`) with an NFS mounted file system

A remote login (`rlogin`) or a remote shell (`rsh`) must be used to issue FileServ commands when not logged into the FileServ host.

Interface to VolServ

After both FileServ and VolServ have been installed and initially configured, drive pool and association of a MediaClass group with a library must be established within VolServ for FileServ communications.

Drive Pools

A drive pool is composed of a single drive or group of drives that have the same capability. Drive pool names used within FileServ **must** exist in VolServ before any data transfer operations can be performed.

A drive pool is assigned a unique name to identify the drives or system using the drives.

The system administrator can either specify a drive pool name when adding DataClass group in FileServ, or can use the default drive pool name found in the system parameter `VS_DEFAULT_DRIVEPOOL`. The system administrator can also limit a set of drives to a set of clients by creating different drive pools and specifying the names of each drive pool for each DataClass group associated with those users.

During FileServ installation, a drive pool name is created containing all drives. The drive pool name is then added to VolServ. The generated drive pool name is under FileServ's `VS_DEFAULT_DRIVEPOOL` parameter. If this name is not configured in VolServ, all requests for data operations fail.

Several FileServ commands allow the user to specify a drive pool using the `-v` option. If a drive pool name is specified, it must either reside or be created within VolServ before any FileServ commands can be executed.

Associate MediaClass with Library

On SGI systems, there are fifty four MediaClass names. Each MediaClass has the following form:

`FileServID_mediatype_mediaclasstype`

There are ten possible values for FileServ ID: F0 through F9.

The following `mediaclasstype` exist:

- ADDBLANK or ADDBLNK
- CHECKIN
- IMPORT
- DATA
- MIGRATE
- REMOVE

An example of a valid MediaClass name is F0_D2S_ADDBLANK.

The FileServ installation script creates each of the MediaClass names as a VolServ MediaClass group. After the FileServ installation, the VolServ system administrator must associate the appropriate FileServ default MediaClass names with the appropriate library.

An example of valid MediaClass names that must be associated with a library are as follows:

- `F0_D2S_ADDBLANK`
- `F0_D2S_CHECKIN`
- `F0_D2S_IMPORT`
- `F0_D2S_DATA`

- `F0_D2S_MIGRATE`
- `F0_D2S_REMOVE`

Register the Configuration

Before FileServ can operate properly, the physical configuration of the system must be defined within FileServ. This procedure is accomplished during or after FileServ installation.

After installation, the system administrator is responsible for associating all the drive devices configured in VolServ to FileServ.

The `fsconfig` command provides hardware component configuration.

Note

For more information on the `fsconfig` command, refer to the *Command Reference* book.

Define DataClass Groups

To use FileServ to either migrate or archive data, directories in various file systems must be associated with sets of parameters that govern data migration characteristics. The migration parameter set is known as a DataClass group.

A DataClass group is created and defined through the `fsaddclass` command or **`Define Classes`** screen.

The DataClass parameters control the following:

- Allocation of media.
- Number of copies of file data that are stored.
- Application of the data storage.

- Truncation policies for files and media types associated with the DataClass group.

Note

For more information on the `fsaddclass` command, refer to the *Command Reference* book.

Associate Directories

After a DataClass group is defined, it can be associated with directories in a file system so that files written to that directory structure have their file data migrated according to the parameters defined for the DataClass group.

Caution

Never associate the directories that contain the FileServ executables with a class!

DataClass group association is implemented through the `fsaddrelation` command or `Define Class/Directory Relations` screen. Only files written to directories that are associated with a DataClass group are subject to the migration policy application of FileServ.

These files are stored using the `fsstore` command.

Note

Associated directories and files are unavailable when FileServ is inactive. ADIC recommends that directories that contain applications or login files not be associated.

Media Under FileServ Management

Before data can be copied from disk to tape, the tape media must be in place. The media should be properly labeled and entered into a storage subsystem by a VolServ operator before entering media into the FileServ database.

The `fsmedin -b` command is used to add blank media into the FileServ storage subsystem.

The total *quantity* can be any number; a default value of 99 is used if *quantity* is not specified. The total *quantity* should be restricted to a number equal to or less than that entered into an archive following VolServ installation.

The *mediatype* name should match the type of assets associated with the library located at your site. Formatting each tape requires about 1 minute for DLT and 3590 media, 6 minutes for D-2S, 15 minutes for D-2M, and 35 minutes for D-2L. Use the command below to format a tape.

```
% fsmedin -b -q quantity -t mediatype
```

If D-2 tape formatting is preferred, time is saved by changing the `$FS_HOME/sysparms/fs_sysparm_override` file. The `FS_MAX_ACTIVE_FORMATS` parameter has a default value of 1, which means only one tape drive can be used at one time for formatting. This value can be changed using an editor to a number equal to the number of tape drives available for formatting.

Create Migration Policies

Migration policies initiate movement of file data from disk to tape.

Tip

ADIC recommends that migration policies be applied on a routine basis. Periodic application of migration policies is best accomplished through the use of a crontab, to invoke many routinely executed UNIX commands.

The following types of policy can be run:

- Store data to tape (based on DataClass parameters).
- Remove data from disk (based on disk space)

The `fspolicy -s` command is used to invoke the policy to store data to tape. Eligibility for storage is based on the DataClass parameters of *class*.

```
% fspolicy -s -c class
```

The `fspolicy -t` command is used to invoke the policy to truncate data from disk after it has been stored to tape. Eligibility for truncation is per file system *filesystem* and based on the file attributes, such as last access date.

```
% fspolicy -t -y filesystem
```

Routine Processing

Part of the FileServ system routine processing includes:

Store and Truncate Files

Policies for storage and for data truncation are run as part of routine system processing. The standard method of running routine operations is to have the operating system `cron` process execute the operations. A `root` generated `crontab` is read and registered with the ongoing `cron` process. The application of migration and truncation policies is handled as part of this process. The `cron` file contains many routinely executed by UNIX commands.

Database Maintenance

The database used by FileServ is an intricate part of the software. Access to the database is controlled through a process called the Lock Manager. The Lock Manager is started by the FileServ system administrator. The startup script is called `lm_start` and is located in the `$FS_HOME/util` directory. If the Lock Manager is not running, FileServ will not start.

To prevent data loss and improve FileServ performance, periodic database maintenance must be performed.

FileServ database maintenance is part of routine system processing. FileServ uses a RAIMA database to store and track information about the files under FileServ. The RAIMA database is embedded within FileServ and automatically starts when FileServ starts.

Periodic database maintenance is required to prevent data loss, improve FileServ software performance, and simplify recovery procedures if needed.

The following three utilities should be run on a periodic basis:

- $FS_HOME/internal/chk_point/chk_point.pl
- $FS_HOME/util/checkDBALL
- $FS_HOME/util/keyBuildAll

chk_point.pl

The `$FS_HOME/internal/chk_point/chk_point.pl` script checkpoints the RAIMA database (database recovery is faster and more reliable when regular checkpoints are taken).

Checkpointing of the database is **not** automatically established during FileServ installation. Therefore, checkpointing must be placed in the `crontab` file by the system administrator and executed according to parameters of `crontab`.

Also, checkpointing cannot be accomplished without having Perl installed. Checkpointing should be done during low usage times.

Since checkpointing is a snapshot of the RAIMA database, taking regular checkpoints makes recovery of the database faster and more reliable.

When the checkpoint script runs, it `tars` the entire contents of the FileServ database to a separate file system. After this process is completed, it zeros out the journal file. If the journaling file system becomes full, processing stops. Therefore, it is important to checkpoint the database regularly through a `cron` process.

Checkpointing can be written to the host computer disk or table, but these files must **not** be placed under FileServ's control.

checkDBALL

The `$FS_HOME/util/checkDBALL` script checks the database consistency by validating the position of each record occurrence and checking the integrity of deleted chains. The consistency check verifies that the member and owner record types are valid, the membership count is current, and the doubley linked lists are properly formed.

The script also validates the existence of the key values associated with each record and the key field in the data files. For each key in the key file, the script reads the associated record and checks to ensure that the key's data field contents match that stored in the key file. Finally, the `checkDBALL` script performs a timestamp consistency check on the database.

The `checkDBALL` database script should always be run after a host platform crash because FileServ may have been performing a database transaction when the crash occurred.

The table below shows the suggested database maintenance intervals for each database utility.

Note

Although the terms “Normal Usage” and “Heavy Usage” in the table header are somewhat subjective and may have different definitions at different sites, in general, the `checkDBALL` script and `keyBuildAll` script should be run every 100,000 storage, retrieval, modification, creation, and/or deletion of files within the FileServ controlled DMAPI file systems.

Database Maintenance Script	Normal Usage	Heavy Usage
`$FS_HOME/internal/chk_point/chk_point.pl`	Daily	Daily
`$FS_HOME/util/checkDBALL`	Monthly	Bi-weekly
`$FS_HOME/util/keyBuildAll`	Monthly	Bi-weekly

Although FileServ must be terminated, the Lock Manager process must be active, to run the `checkDBALL` script. The script may be run by either user `fsadm` or user `root`.

ADIC recommends that the output of the `checkDBALL` script be redirected to a temporary file so that an analysis can be performed after the script completes it’s execution. This is accomplished as shown below:

```
# $FS_HOME/util/checkDBAll > temp_file_name
```

After the script completes it’s execution, the `temp_file_name` file should be analyzed for any errors.

keyBuildAll

The `$FS_HOME/util/keyBuildAll` script rebuilds all the key files for the RAIMA database. Rebuilding the file keys is a two step process that also rebalances the B-tree structure of the database.

To rebuild the file keys, follow the steps below:

Step 1. Reinitialize the file.

Step 2. Sequentially reads each record is from each data file record and recreates each key file is from the record contents.

Run the `keyBuildAll` script if one of the following occurs:

- If any errors were detected by the `checkDBALL` script.
- Periodically, based on system usage. (check the system usage in the previous table)

For instructions on file recovery or UNIX to FileServ comparison, see “Audit Database” on page 4-17.

Database Utilities

There are 19 database utilities that allow you to add, modify, delete and query a specific FileServ table.

The table utilities are: `activefl`, `blockpos`, `cfgdir`, `classdef`, `classdir`, `devdb`, `dirdb`, `exitinfo`, `externmed`, `fileattr`, `filecmnt`, `filecomp`, `filexpir`, `keytest`, `medbpos`, `mediadir`, `transdb`, `trashcomp`, and `trashname`.

The utilities that operate on specific file systems are: `blockpos`, `dirdb`, `fileattr`, `filecmnt`, `filecomp` and `filexpir`.

Note

The one table utility that does not work like the others is the keytest utility.

All of the utilities, with the exception of `keytest`, have two modes:

- Command Line Mode.
- Interactive Mode.

Both modes are described below.

For both Command Line Mode and Interactive Mode, an additional parameter called the `device key` is required before running the utility. This parameter must be the first parameter given to the command

Step 1. To determine the `device key`, consult the `devdb` table. An entry or output from the `devdb` utility may look similar to the output shown below:

```
dvdmhdl              :099c17edal7e8fd1
dvdmsz               :8
device key           :4
dvpath               :/test2
```

Step 2. Assume that you now want to check files on the `/test2` migration file system.

First, you check the `devdb` table for the entry containing `/test2`. Notice there is a field called "`device key`" associated with `/test2`. This is the value you will use for device specific table utilities that are associated with the `/test2` file system.

Step 3. To look at the `fileattr` entries for the files on `/test2` and to send the output to a file called `test2.files`, enter the following command:

```
# fileattr 4 -ftest2.files
```

Examine the file at your leisure.

Command Line Mode

Command line mode queries only the specified table. In most cases you need to enter only the utility name to call the query. The utility queries the database for all records in that specific table. The results appear on the screen, or you can redirect the results to a file using the standard UNIX redirection command.

Options for most of the utilities are listed below:

Option	Description
`-f`*filename* **Note:** Do not put a space between the `-f` and the *filename*.	Send all output to the specified *filename*.
-h	Display a help screen
-i	Run in interactive mode

Option	Description
-k *key*	Query for a specific record key. A working knowledge of the database internals is often needed to provide the proper key. Each table utility has a different requirement for the *key*. For additional information, see the HELP screen.

Interactive Mode

To start a utility in Interactive Mode, include the `-i` option on the command line. If the `-i` option is used, FileServ ignores the `-k` *key* option.

In Interactive Mode, a menu displays with options valid for the specified utility. The following table describes the available options.

Options	Description
a	Add a record. You can add a record to any table. You will have the opportunity to change every field and enter the record as you like. In some tables, key fields are generated automatically and are not part of the add record process. If you need to change a key field, this can be done under the modify process.
d	Delete a record. To delete a record, knowledge of specific keys is required. Queries may need to be performed in order to determine the specific key the delete process requires before deleting a record. Once the record is found, it is displayed for confirmation before deletion. Be aware that in some tables this will result in more than one record for deletion.
e	End this program.
I	Retrieve record by class id.

Options	Description
`i`	List class ids.
`k`	List class indices.
`K`	Retrieve record by class index.
`m`	Modify a record. Modifying a record requires a specific key that corresponds to the record to be modified Once the record is found, any field can be modified including key fields. Modifying a key field can result in a duplicate key for the table. In such cases, the modification will fail.
`q`	Query a table. Each table has different fields that are used for queries. There is no specific field that each table is queried on. Some queries allow for a sub menu, while other table queries have several fields that can be queried.
`r`	Record count.
`S`	Set output destination.
`v`	View all records.

In addition, there is always an option to query on one or more unique keys.

When entering an option, you may enter as many options on the command line as you like. However, only the very first character is accepted as the command. If that is a space, the command is in error and nothing is done.

Frequency of Use

How frequency you should use these utilities depend on system usage. It may be advantageous to make checkpoints more often when FileServ database activity is heavy. Such activity includes:

- Addition or deletion of many migration directories.

- Heavy storage, retrieval, modification, creation, or deletion of files in migration directories.

The advantage to taking extra checkpoints is that recovery from database corruption is much faster and more reliable. However, a checkpoint done during heavy system use takes longer and can create serious performance degradation for FileServ processes. By weighing these factors, the database system administrator and the FileServ system administrator must decide the time of day and frequency for each database maintenance operation.

No matter how busy a system is, failure to perform database maintenance on a regular scheduled interval increases the chances of system downtime and loss of data.

Recover Database

If you have total database loss or corruption, a database recovery will be necessary to recover the files under FileServ.

Step 1. To recover the database, run the `restore.pl` script located under `$FS_HOME/internal/chk_point`.

The amount of time required for a recovery depends on how large the FileServ database is and how large the `journal` file is. If the checkpoints are made often, the recovery process will go quickly.

Step 2. After the `restore.pl` script completes, as `root` type `$FS_HOME/util/fs_recover` without parameters.

```
# su - root
# $FS_HOME/util/fs_recover
```

Run this command **only** if the archived file systems are newer than the restored database and the journal files have been preserved. The `restore.pl` command will not replace the old journal, so no other preparatory work should be needed if the journal file has been preserved. The journal file is located in `$FS_HOME/journal/fsdbjrnl`.

Step 3. On the other hand, if the database was lost, but the current `journal` file is available, the database can be brought up-to-date by running `fs_recover`.

Trashcan Cleanup

Another command that must be routinely run is `fsclean`. The `fsclean` command purges the FileServ trashcan information.

Caution

Using the `fsclean` command eliminates the ability to recover files from the trashcan with the `fsundelete` command!

Precautions must be taken when using the `fsclean` command. The trashcan serves as the repository for references to media data that is deleted from the disk or modified. Each time a file is modified, its tape copy becomes invalid. An entry is created in the trashcan for the older version of the file. Also, when files (containing current file copies on media) are removed (UNIX `rm`), the trashcan receives an entry for each removed file.

Because of asynchronous processing, an `fsclean` performed immediately after file removal may not completely clean the media.

Run a `fsmedinfo -l` report to verify all files deleted from media. If files still exist, rerun the `fsclean` command.

Note

Execute the `fsundelete` command twice to make sure all your required files have been undeleted.

The `fsundelete` command allows recovery of removed or modified copies of files as long as an entry exists in the trashcan. Only the primary copy is recoverable with the `fsundelete` command. After `fsclean` is run for media, all files on that media referenced in the trashcan are no longer recoverable with `fsundelete`.

Startup and Shutdown

The following startup modes exist for FileServ:

- **Normal**. A normal termination allows a request to continue to a known processing state. When a normal termination is unsuccessful, contact technical support personnel to assist with the correction. If termination exits with a message that the system abnormally terminated, contact the technical support personnel.
- **Contingency**. A contingency start is only recommended when a normal start fails.

Both types of startup are described below.

Normal Startup

A normal startup is performed following a graceful termination of FileServ. A normal FileServ startup performs the following tasks:

- Reinitializes the system parameters.
- Performs command recovery processing using the recovery processing information retained after termination.

To normally start or terminate FileServ, the system administrator runs the `FileServ` command. This command must be issued from a login to the host computer.

This command can also be included in a script that is run automatically at the time of the host computer initialization. The `FileServ` command will automatically starts FileServ each time the server is restarted.

The licensed subsystems must be configured before startup or FileServ will fail to start.

The following messages display at the command line where the `FileServ` command is run to indicate a successful initiation*:*

```
FS0338 FileServ: Request accepted
FS0277 FileServ complete: FileServ software is
running
```


Recovery Processing

During FileServ processing, software activity is continually recorded in recovery processing files. These files are only used during a normal startup. It is important to always attempt a normal start after any termination of FileServ.

Caution

If files that are migrated to media are actively being transferred to disk at the time an abnormal software termination occurs, a contingency start may corrupt the file data.

A normal start resets the active files to the original state before starting the transfer. Files that are on media only and are being retrieved are reset to being on media only. The request that was active at termination must be reissued by the requestor.

Contingency Startup

A contingency start is only recommended when a normal start fails. With the exception of initial startup at installation, **never** attempt a contingency start without first attempting to perform a normal start of FileServ.

Note

A contingency start is used for first-time Initialization of FileServ. For more information, refer to *Installing FileServ.*

A contingency start initializes certain log files and internal directories but does not perform recovery processing.

The recovery files cannot be used after a contingency start is issued. If the user needs to save the recovery files for review at a later time, move these files to another directory. Do not rename the recovery files because the software deletes all files found in the recovery file directories.

These activities allow FileServ to return to a known processing state when a normal start cannot be performed.

Caution

All files in the recovery directory are deleted during a contingency start. To retain files, move all recovery files to a different directory.

Step 1. Run the `FileServ -c` command.

```
% FileServ -c

FS0290  CAUTION! Contingency startup deletes recovery
processing files.
FS0284 FileServ contingency startup requested.
FS0290 Caution! Contingency startup deletes recovery processing
files:
FS0293...
FS0293...
FS0294 Are you sure you wish to continue? (y/<n>):
```

Step 2. Press <RETURN> to cancel the contingency startup or enter **Y** for yes.

Normal Shutdown

To ensure an orderly shutdown of FileServ, any processing must be terminated gracefully.

All components must be shut down in the sequence as follows:

Step 1. Shut down FileServ.

Step 2. Shut down VolServ.

Step 3. Shut down Ingres database software.

Step 1. Run a system-wide broadcast message that states FileServ will be terminated. Use the UNIX `rwall` command to issue a message similar to the one below to all hosts and their clients on the network.

```
FileServ services will be terminated in
n minutes
```

Step 2. Repeat the broadcast before termination.

Step 3. Run the `FileServ -t` command. The following message is returned.

```
FS0285 Termination requested by user.
FS0294 Are you sure you wish to
continue? (y/<n>):y
```

Files being actively *transferred to media*, or *from media* are completed. File transfer requests awaiting resources are terminated and a request aborted status is returned to the user.

All files are closed and cleanup is performed to ensure that the system is in the proper state to allow an orderly restart.

Any media introduction or removal operations in progress are interrupted. The requests resume and are completed when a subsequent normal startup is performed. The configuration states for the storage library components are not changed during termination.

Step 4. Perform an orderly system shutdown only after FileServ has terminated.

Step 5. Enter the following command to make sure that unwanted processes are not still running:

```
# ps -elf | grep fs_
# ps -elf | grep fsadm
```

Automate Software Termination

FileServ terminates if the software is unable to perform a `commit` or `rollback` operation on the database for a resident FileServ process. Error messages are sent to the system logs and console to indicate that FileServ is terminating because of a database software error. When the database software problem is resolved, bring up FileServ.

Managing System Configuration

For the proper execution of FileServ commands and operations, the physical storage library configuration must be maintained in the FileServ system and parameter files. The table below lists the tasks that the FileServ system administrator can perform on the system and files.

System Tasks	Page
Modify System Parameters	1-27
Reconfigure System Devices	1-47
Change Component State	1-53
Add and Delete Drives	1-55

Modify System Parameters

FileServ employs global parameters that define the system. These parameters are located in files contained in the `$FS_HOME/sysparms` directory, where `$FS_HOME` is the directory where FileServ is installed.

FileServ parameter files can be edited to customize the software for your specific installation. For instructions, see "Edit System Parameters" on page 1-46.

The table below lists the configurable parameters:

Configurable System Parameters	Page
General Parameters	1-28
System Administration Parameters	1-28
Default Parameters	1-36
File Retention Parameters	1-39
VolServ-specific Parameters	1-40

General Parameters

The `fs_sysparm` file contains parameters that control general FileServ activities. The parameters in this file are grouped into the following categories:

Category	Description	Page
System Administration	Used to tune system performance and media usage.	1-28
FileServ Defaults	Used with FileServ commands that allow default value.	1-36
File Retention	Used by the file comment keyword search and file expiration daemons.	1-39
Installation	Established at the time of system installation.	1-41
Software Maintenance	Modified as required for software maintenance.	1-43

System Administration Parameters

The system administration parameters in the table below can be changed to tune system performance and media usage. FileServ must be cycled (using the `FileServ -t` and `FileServ` commands) to pickup changes to these parameters.

Parameter	Default	Definition
`NOMINAL_FILE_SIZE`	5000000	Nominal file size in bytes to estimate how many bytes of user data fit on the remaining space on the media.

Parameter	Default	Definition
FILE_LBL_FMT_TAPE	adic00000001	Defines the way data labels are written to tape devices. adic00000001 is ANSI standard and writes data with tape marks between the header labels and data, and between data and trailer labels. adic00000002 leaves off tape marks. This is faster, but is not ANSI standard.
FILE_LBL_FMT_DLT	adic00000003	Defines the way data labels are written to RSP-2150 devices. adic00000001 is ANSI standard and writes data with tape marks between the header labels and data, and between data and trailer labels. adic00000003 leaves off tape marks. This is faster, but is not ANSI standard.
FILE_LBL_FMT_RSP	adic00000003	Defines the way data labels are written to RSP-2150 devices. adic00000001 is ANSI standard and writes data with tape marks between the header labels and data, and between data and trailer labels. adic00000003 leaves off tape marks. This is faster, but is not ANSI standard.
FILE_LBL_FMT_8590	adic00000003	Defines the way data labels are written to RSP-2150 devices. adic00000001 is ANSI standard and writes data with tape marks between the header labels and data, and between data and trailer labels. adic00000003 leaves off tape marks. This is faster, but is not ANSI standard.

Parameter	Default	Definition
DEFAULT_MEDIA_TYPE	DLT	Default media type used with commands with optional media types.
DEF_MED_SPC_3480	220000000	Default tape length of a 3480 cartridge in bytes.
DEF_MED_SPC_3490	400000000	Default tape length of a 3490 cartridge in bytes.
DEF_MED_SPC_3490E	800000000	Default tape length of a 3490E cartridge in bytes.
DEF_MED_SPC_D2SM	25000000000	Default tape length of a D-2 small cassette in bytes.
DEF_MED_SPC_D2MD	75000000000	Default tape length of a D-2 media cassette in bytes.
DEF_MED_SPC_D2LG	165000000000	Default tape length of a D-2 large cassette in bytes.
DEF_MED_SPC_CTIII	10000000000	Default tape length of a DLT cartridge in bytes.
DEF_MED_SPC_CTIV	20000000000	Default tape length of a DLT cartridge in bytes.
FS_EOT_SIZE_RESET_ FACTOR	0.5	Fraction of space available that will be added to current write position when physical End-Of-Tape (EOT) is detected for systems utilizing 3490 media. Smaller fractions should be used when fewer mounts and performance for robotic utilization is a priority. Higher fractions should be used when tape utilization is a priority. Valid range is 0.0 - 1.0. It is recommended that this value not be modified prior to contacting technical support.

Parameter	Default	Definition
FS_MAX_ACTIVE_ FORMATS	1	Maximum number of drives allowed to perform tape formatting at one time.
FS_MAX_ACTIVE_ MEDCHECKS	1	Maximum number of drives allowed to perform media checking at any one time.
STORE_LIMIT_NOTICE	1	If FileServ is started up when the storage used is within this number of GB from the storage limit, notice is sent to FS_OWNER_ID.
FS_CALLOUT_SLEEP_INT	2	A file can be accessed by one process at a time. This parameter represents the time to wait (in seconds) before the next retry if callouts are suspended because of a file busy. (For tuning, refer to note in parameter MAX_DMON_SUSP_ERRORS.)
MAX_DMON_SUSP_ERRORS	60 two-second retries	Maximum number of retries in response to a suspend failure. The command sleeps between retries for the number of seconds specified in FS_CALLOUT_SLEEP_INT. A suspend failure can occur for a file if: • The file has already received a callout. • The maximum number of suspends is already reached.

Parameter	Default	Definition
(continued)		The FS_CALLOUT_SLEEP_INT and MAX_DMON_SUSP_ERRORS parameters can be tuned, if recurring failures of multiple copies of files are noted in the system logs. This situation can occur in a backup environment where the primary and backup copies of large files are both stored at the same time by issuing the `fspolicy` command. The default can expire on the second copy of a large file while the first copy is written to media. **Note:** Run `fspolicy -w` to rebuild candidate files. For additional help in tuning these parameters, call ADIC technical support.
`CONNECT_SLEEP_TIME`	1	Time to wait (in seconds) before the next retry if the IPC Connect request fails.
`CONNECT_RETRIES`	120	Number of retries on IPC connection failure. This is set by the factory and is changed only by technical support personnel.
`MAX_READS`	100	Number of retries on IPC receive failure. Set the `MAX_READS` and `MAX_WRITES` parameters to the same number. This is set by the factory and can be changed only by technical support personnel.

Parameter	Default	Definition
MAX_WRITES	100	Number of retries on IPC send failure. The MAX_READS and MAX_WRITES parameters are set to the same number. This is set by the factory and is changed only by technical support personnel.
FS_DB_RETRY_COUNT	3	Number of retries on database services failure because of table-busy errors.
DMON_POLL_TIME	10	Time to wait (in 1/100 seconds) on select poll for events on IPC queue. This is set by the factory and can be changed only by technical support personnel.
MAX_RETRIEVE_RETRY_ COUNT	2	Maximum number of retries for any retrieve operation. Each retry is performed on a different file copy.
LOBLK_THRESHOLD	85	Percentage of used disk space when reached initiates the *overflow* utility to reduce the level of used disk space down to the HIBLK_THRESHOLD value.
HIBLK_THRESHOLD	84	Minimum level of used disk space that the `overflow` utility maintains.
MAX_TAPE_TO_TAPE _ALLOC_TIME	60	Maximum amount of time (in minutes) that a Medcopy request waits for resources. When this threshold is exceeded, the request fails. (Valid range is 1 through 10000.)

Parameter	Default	Definition
STARVATION_PERCENT	50	Percent of MAX_TAPE_TO_TAPE_ALLOC_TIME variable in which resources will be obtained and not released until all resources have been secured or the time limit has been exceeded. (Valid range is 1 through 100.)
FS_NICE_VALUE	n	If enabled, the FileServ process and database resident process runs with this *n* value. Refer to set priority (2) for the value range of the system. To enable, enter a value. To disable, set the value to "0."
FS_VS_QUEUE_XXX_THRESHOLD	variable	Storage request threshold value when surpassed, resource requests are queued in the FileServ system instead of the VolServ system. The Value is 3x the number of drives located in the system. For each media type, there is a separate entry required, e.g., FS_VS_QUEUE_D2S_THRESHOLD. If the value is changed, it is recommended that the sum of all FS_VS_QUEUE_xxx_THRESHOLD values not exceed 100.
FS_THRESHOLD_INC_NUM	5	Threshold increment number variable. The value for the drive failure level when an access to a drive fails.
FS_THRESHOLD_DEC_NUM	1	Threshold decrement number variable. The value for the drive failure level when an access to a drive is successful.

Parameter	Default	Definition
FS_DRIVE_ERR _THRESHOLD	20	Threshold value when equalled or exceeded results in drives being taken offline.
FS_MAX_FILES_PR_3480	2000	Maximum number of files allowed on a single 3480 cartridge tape.
PERCENT_FULL_TO _MIGRATE	95	Percentage value used to check against to determine if media can be migrated.
File Copy Block Factors	device-dependent	Size of the following *blocks* is a calculated optimum for each device type.
FS_DISK_BLOCK_FACTOR	10	Number of disk blocks used per transfer when reading/writing the disk. Parameter is normally commented out. Use this parameter with systems that have limited memory. System performance is degraded during file copies. Contact technical support before activating.
FS_TAPE_BLOCK_FACTOR	1	Number of tape blocks used per transfer when reading/writing a tape. Parameter is normally commented out. Use this parameter with systems that have limited memory. System performance is degraded during file copies. Contact technical support before activating.
FS_RSP2150_BLOCK _FACTOR	4	Number of 2150 blocks used per transfer when reading/writing the 2150. Parameter is normally commented out. Use this parameter with systems that have limited memory. System performance is degraded during file copies. Contact technical support before activating

Parameter	Default	Definition
CLEANUP_PROCESSING	100	Number of files processed per transaction in the **f**`sclean` command. If this value is set higher than 100, the results can be memory allocation problems and increased processing time. 100 is the recommended value.
FS_PRINTER_PATH	/usr/bin/lpr	Default printer used for printing reports from the GUI.
MDM_POSITION_VALIDATION	YES	Enables validation of the tape position prior to the first file in a file set.

Default Parameters

The FileServ default parameters are used with FileServ commands that allow defaults (`fsaddclass` and `fsmodclass`). To change a default, modify this file, cycle FileServ, then retry the command.

Review these parameters to verify that the applicable defaults are site specific. Because the defaults are used when no specific value is indicated in a command, these defaults are set to values that are most often used. This allows increased use of the defaults in the commands. Modify those values that are contrary to a site's specific operations. The following example illustrates the type of issues for which it is beneficial to modify the FileServ defaults.

If the system administrator does not log on as `root`, change the `CLASS_USERID` to default to the user ID of the user who is available for mail notification of actions that are required by FileServ.

Parameter	Default	Definition
`CLASS_USERID`	fsadm	Default userid. Any valid userid can be used. This is the default for the E-mail Notify ID.
`CLASS_ACCTNUM`	12345	Default account number. One to five alphanumeric characters can be entered.
`CLASS_SCODE`	NONE	Default security code. One to four alphanumeric characters can be entered.
`CLASS_SOFTLIMIT`	8	Default warning limit for the number of media in a class. The warning is issued when the number of media are allocated to the class. The default number can be changed when setting up the class.The number can also be modified for an existing class with the `fsmodclass` command.
`CLASS_HARDLIMIT`	10	Default hard limit for a class. Additional media is not allocated for the class when this limit is reached. The default number can be changed when setting up the class by using the `fsaddclass` command. The number can also be modified for an existing class with the `fsmodclass` command. The classes value must be greater than the value for the CLASS_SOFTLIMIT.
`CLASS_MTIME`	10	Default minimum time (in minutes) since a file was last accessed. A file is eligible for policy application (store or cleanup) after mintime. The number can also be modified for an existing class with the `fsmodclass` command.

Parameter	Default	Definition
CLASS_DEF_MEDIA_TYPE	DLT	Default media type (D2S, D2M, D2L, DLT, 3480, 3490, 3490E, 3590,8590). Depending on the platform used or a manual system, modify the default to media used. The number can also be modified for an existing class by using the fsmodclass command.
CLASS_FILE_SPAN	N	Default flag to allow file spanning media (**Y** or **N**). The value can also be modified for an existing class by using the `fsmodclass` command.
CLASS_FILE_CLUSTER	N	Default flag to allow file clustering (**Y** or **N**). The value can also be modified for an existing class by using the `fsmodclass` command.
CLASS_MAX_COPIES	2	Maximum number of copies allowed, including the primary copy. A maximum number of two copies is allowed for the current version of FileServ. The number can also be modified for an existing class by using the `fsmodclass` command.
CLASS_DEFAULT_COPIES	1	Total number of copies stored to media for each file in a class. Must not exceed CLASS_MAX_COPIES. The number can also be modified for an existing class by using the `fsmodclass` command. The default can be modified for a file by using the `fschfiat` command, although the number must not exceed CLASS_MAX_COPIES.

Parameter	Default	Definition
CLASS_FILE_CLEANUP	P	Default file cleanup action (I or P). When a file is stored, cleanup can occur immediately (I) after storing the file or at policy application (P). The value can also be modified for an existing class by using the `fsmodclass` command. If the *sysparm* value is not set to I or P, FileServ defaults to P.
CLASS_MEDIA_CLEANUP	S	Default media cleanup action (C or S). When media becomes logically blank, the media can return to the class blank media pool (C) or to the system blank media pool (S). The value can also be modified for an existing class with the `fsmodclass` command. If the *sysparm* value is not set to C or S, FileServ defaults to C.
FS_EPSON_LABEL_PRINTER		Default printer for printing external media labels. Specifies the Epson printer that generates the external top media labels. To enable, contact technical support.
CLASS_DRIVEPOOL	fs_F0drivepool	Default drive pool (up to 16 characters). Usually set to the same value as VS_DEFAULT_DRIVEPOOL.

File Retention Parameters

The following list shows parameters that are used by the file comment keyword search and file expiration processes. To change a value, modify this file, cycle the user interface, then retry the command.

A file will be retained on the disk based upon the DEF_FILE_RETION_PERIOD or the FILE_RETENTION_PERIOD value. The lowest value will determine a file's true retention period.

Parameter	Default	Definition
ADVANCE_NOTIFICATION_ PERIOD	30	Number of days in advance the DataClass group manager is notified of the impending expiration of files located in a DataClass group. Valid range is from 1 and 32767 days.
DEF_FILE_RETENTION_ PERIOD	0	Time a file is kept on the disk. Valid range is from 0 to 32767. If value is set to 0, the file will be kept on disk indefinitely. If a value from 1 to 32767 is used, the file will be retained for the number of days entered beyond the last access date.
FILE_RETENTION_PERIOD	0	Time a file is kept on disk. Valid range is from 0 to 32767. If value is set to 0, the file will be kept indefinitely. If a value from 1 to 32767 is used, the file will be retained for the number of days entered beyond the last access date.

VolServ-specific Parameters

The following parameters are specific to the VolServ system. FileServ must be cycled to pick up changes to these parameters.

Parameter	Default	Definition
FILESERV_ID	F0	A two-character unique identifier used to differentiate the FileServ systems connected to the same VolServ host. This identifier is the prefix for all MediaClass names and drive pool names.
VS_HOSTNAME	XXXXXXX	VolServ hostname to be used by this FileServ system.

Parameter	Default	Definition
VS_PROGRAM_NUMBER	XXXXXXX	If enabled, the VolServ program number to be used by this FileServ system. To disable, comment out the parameter.
VS_DEF_QUANTITY	99	Default media quantity for entry/exit port operations. (Range 1-99)
VS_DEFAULT_DRIVEPOOL	fs_F0drivepool	Default VolServ drive pool used for retrieves. This drive pool contains all FileServ-configured drive components.This drive pool is used for media with no DataClass group. Defined in terms of FILESERV_ID.
VS_DIR	XXXXXXX	This parameter should point to the VolServ environment for the fspic utility to run logoffcomps.

Installation Parameters

The following list shows the system parameters that are to be established at the time of system installation. Use the FileServ installation script for this purpose.

As a general rule, do **not** modify these rules unless the system is completely reinitialized and reinstalled.

Parameter	Default	Definition
FileServ_LICENSE_STRING		FileServ license string.
FS_ADMIN_DAEMON_ID	2	FileServ daemon number (0-3) used for administrative activities.
FS_DATA_DAEMON_ID	3	FileServ daemon number (0-3) used for data activities.

Parameter	Default	Definition
FS_DATABASE	fsdb	FileServ database name (1-20 chars). The database maintenance scripts use **fsdb** as the FileServ database name. If this parameter is modified, these scripts must also be modified.
FS_CONSOLE	/dev/console	Identifies the device path for the FileServ console.
FS_DEFAULT_SUBSYSTEM	V0	Default component ID only for the `fsmedin`, `fsmedout`, and `fsqueue` commands when the user does not specify the subsystem on the command.
These parameters below set (UGO) permissions, owner, and group ID for any files created by FileServ.		
FS_FILE_MODE	511 Decimal value of octal permissions 777.	UGO permissions for files created by FileServ.
FS_FILE_GROUP	adicadm	Group ID for files created by FileServ.
FS_OPR_GROUP	adicopr	Group ID for fs operators.
FS_OWNER_ID	fsadm	Owner ID for FileServ files and database owner's userid.

Software Maintenance Parameters

The following list shows parameters that can be modified as required for software maintenance. FileServ must be cycled to affect the changes. These parameters are used to enable and disable different types of message logging. These logs can be used for debugging purposes. To enable logging, remove the comment line indicators from the parameter(s) and recycle FileServ.

Parameter	Default	Definition
COMMAND_LOGGING	y	Enables logging of user commands and status to the fs_hist_21 file.
COMMAND_INFO_DETAIL	y	Removes the message number, priority, and request identifier from the status returned to the user.
COMMAND_EXTRACT_HEADER	n	If enabled, removes the message number, priority, and request ID from the command status back to the user. Does not effect logging.
FS_TRACE_MASK	o	Controls generations of trace logs to the /FS_HOME/tracelogs directories. Set Mask according to which trace logs generate. Use individually or in combination. • k = KRPC messages • i = IPC messages • o = no trace logging performed
FS_TRACE_SIZE	8192	Record size limit (in bytes) for recording in the trace logs.

Media Stats Parameters

The following list shows parameters that can be modified as required, to collect media statistic information for each site. FileServ must be cycled to pick up changes to these parameters.

Parameter	Default	Definition
TSC_LOG_MASK	e	Enables or disables various priorities of the tape statistic logging. The following values can be used in any combination: • e = error priority (always logged) • h = history priority

Logging Parameters

The following list shows parameters that can be modified, as required, to choose logging levels for each site. FileServ must be cycled for to affect changes. These parameters are used for trace logging.

If trace logging is enabled through the FSLOG_OPTIONS parameter, logs are sent to the trace log files. If trace logging is disabled, then all trace-level logs are discarded.

Parameter	Default	Definition
FSLOG_OPTIONS	e	Enable the trace logging facility to the /FS_HOME/syslog/trace directories. • e = enable trace logging. • - = no trace information is logged.
FS_LOG_LEVEL	YYYYYYYYY	Controls which dynamic technical support logs are sent to the fs_ATAC_11 file. • NNNNNNNN = OFF levels 12-20 • YYYYYYYY = ON levels 12-20

Parameter	Default	Definition
FS_FACILITY	1	Logging facility used for priority 0-7 messages directed to the syslog. The configuration for the syslog is in the */etc/syslog.conf* file. The following facilities can be used with the FS_FACILITY parameter *KEY* (used to direct messages to files): • 0 = LOG_LOCAL0 • 1 = LOG_LOCAL1 • 2 = LOG_LOCAL2 • 3 = LOG_LOCAL3 • 4 = LOG_LOCAL4 • 5 = LOG_LOCAL5 • 6 = LOG_LOCAL6 • 7 = LOG_LOCAL7 For more information, refer to the man page for syslog and syslogd.
FS_LOG_OPTIONS	pc	Log options for priority 0-7 messages that are directed to the syslog. The following options can be used in any combination with the FS_LOG_OPTIONS parameter: • p = LOG_PID • c = LOG_CONS • d = LOG_NDELAY
FS_INT_PERF_LOG	Y	Performance point logging: • **N**= disables logging • **Y**= enables logging

Parameter	Default	Definition
FS_LOG_MASK	ui	The log mask used for priority 0-7 messages that are directed to the syslog. The FS_LOG_MASK parameter allows the “setlogmask” function to be called with modifiable values. These options can be used for the FS_LOG_MASK parameter, either individually or with a’u’ appended to it. • e = LOG_EMERG • a = LOG_ALERT • c = LOG_CRIT • r = LOG_ERR • w = LOG_WARNING • n = LOG_NOTICE • i = LOG_INFO • d = LOG_DEBUG • u = LOG_UPTO() Example: FS_LOG_MASK=e; implies setlogmask(LOG_MASK(LOG_EMERG)); FS_LOG_MASK=ur; implies setlogmask(LOG_UPTO(LOG_ERR));

Edit System Parameters

Configurable system parameters are modified by editing the FileServ system parameters *$*FS_HOME/fs_sysparm file.

Step 1. Edit the fs_sysparm file using a text editor and replace the existing value with a valid new value.

All entries must be in the format: name=value; with no blank spaces around the equal (=) sign and each value terminated with a semicolon (;).

An example for the `CLASS_MTIME` parameter is shown below:

```
rm CLASS_MTIME=10;

CLASS_MTIME=30;
```

Step 2. Recycle FileServ by running the command below:

```
FileServ -t
```

Upon restart, a service utility is called by various processes and routines to establish system parameter values within the FileServ environment. This utility reads the appropriate parameter file to gather the parameter name and its value.

Step 3. Restart FileServ as shown below:

```
FileServ
```

Reconfigure System Devices

Although device configuration of the library hardware is performed during system installation and setup, storage subsystems or drive components can be added to or removed from the FileServ system configuration.

Device configuration involves the allocation of device driver files for all peripherals, including the following:

- Disks
- Libraries
- Tape drives

The UNIX device drivers provide the means to reconfigure devices for the standard peripherals (disks and system console). To reconfigure these components, refer to the applicable platform manual.

Edit Devices with fsconfig Command

After installation, the system administrator can add or remove devices from the configured storage system.

The `fsconfig` command provides hardware component configuration. By specifying the proper options and the accompanying values, hardware components can be added, modified, or deleted from the system to reflect the actual physical configuration.

Note

The `fsconfig` command does not allow duplicate device pathnames in the database.

For more information on the `fsconfig` command, refer to the *Command Reference* book.

The component identifier is a required value to add, modify, or delete hardware. When adding a new drive or subsystem component, the component identifier, component type, and component alias are required values. The component identifier cannot be modified. When deleting a subsystem, the drives must be deleted before the subsystem can be deleted.

When adding or deleting a drive to the system, the system parameter `FS_VS_QUEUE_XXX_THRESHOLD` value located under `$FS_HOME/sysparms` directory must be updated for each media type the drive supports. The entered value is three times the total number of drives in the system. For example, three drives exist in a library system, with a fourth drive to be added. Because the drive supports D-2 media, the system parameter file shows the following values for `FS_VS_QUEUE_XXX_THRESHOLD`:

- `FS_VS_QUEUE_D2S_THRESHOLD = 9`
- `FS_VS_QUEUE_D2M_THRESHOLD = 9`
- `FS_VS_QUEUE_D2L_THRESHOLD = 9`

The value for each supported media type is changed to reflect the addition of the new drive (3x4 drives). If a drive is deleted from the system, the value is changed to reflect the deletion of the drive. If multiple drive types are used, it is recommended that the sum of all `FS_VS_QUEUE_XXX_THRESHOLD` not exceed 100.

Add a Library

Step 1. If a new library is added to your site, add the license string variable to the VolServ `ENVAR` file system parameter file.

Step 2. Add the drives in the new library to the FileServ system configuration list with the `fsconfig` command.

Step 3. Contact technical support for detailed instructions for adding or removing storage subsystems and libraries to your storage system.

Edit Device

Step 1. Run the `fsconfig` command with the **-a** option to add a new component to the ADIC system.

Step 2. Use the appropriate options, as needed. For option information, refer to the *Command Reference* book.

Delete a Device

Step 1. Run the `fsconfig -h` *componentID* command with the `-d` option to delete a device from the configuration.

Step 2. Use other appropriate options, as needed. For option information, refer to the *Command Reference* book.

Generate Report

Step 1. Run the `fsconfig` command.

Use the `-h` option to show the configuration of all component.

Or, use the `-i` option to show the configuration of a specific component.

Step 2. An example of the generated report is shown below:

```
% fsconfig

------------------------------------------------------------
 Hardware Configuration Report Fri Jan 29 09:13:49 1999
   Component ID:     V0
------------------------------------------------------------

  Device pathname:   N/A
     User Alias:     VolServ
  Component Type:    SUBS
     Device Type:    N/A
        Drive ID:    10
     Delay Time:     0
------------------------------------------------------------

 Hardware Configuration Report Fri Jan 29 09:13:49 1999
   Component ID:     V0,10
------------------------------------------------------------

  Device pathname:   /dev/er90/s0
     User Alias:     ER90_DR1
  Component Type:    DRIVE
     Drive Type:     ER90
       Drive ID:     10
     Delay Time:     0
------------------------------------------------------------
```

```
 Hardware Configuration Report Fri Jan 29 09:13:49 1999
   Component ID:     V0,11
------------------------------------------------------------

  Device pathname:   /dev/er90/s1
     User Alias:     ER90_DR2
  Component Type:    DRIVE
     Drive Type:     ER90
       Drive ID:     11
     Delay Time:     1000
```

Drive Compatibility

The `fsconfig` command can configure various drives. The following list shows the FileServ default drive values and the various drives each default value supports along with the media type. When adding a drive, this table should be referenced. The tape media that came with your system may not include all these types.

Drive Names	Supported Drives	Supported Media	Media Types
DLT2000	DLT2000	CompacTape Type III	CTIII
DLT4000	DLT4000	CompacTape Type III	CTIII
		CompacTape Type IV	CTIV
DLT7000	DLT7000	CompacTape Type III	CTIII
		CompacTape Type IV	CTIV
MO5.25	HP2600FX	2.3GB MO disc	MO525
		2.6GB MO disc	
3480	4480	3480 Cartridges	3480
	M2483B (Fujitsu)	3480 Cartridges	3480
	M2483H (Fujitsu)	3480 Cartridges	3480

Drive Names	Supported Drives	Supported Media	Media Types
3490E	4490E	3490 Cartridges* 3490E Cartridges	3480
	M2483N (Fujitsu)	3490 Cartridges* 3490E Cartridges	3480
	M2483ND (Fujitsu)	3490 Cartridges* 3490E Cartridges	3480
8590	8590	8590 Cartridges	8590
* Only blank 3480 media can be written to. On the other hand, 3480 media containing data from a 18-track drive can only be read.			

Change Component State

Component state changes allow for timely maintenance and diagnostics of a library component. When a library component is taken from the `online` (*ON*) state to the `offline` state (*OFF*), it is not available for usual FileServ operations. After the specialized operations are performed, the component must be returned to the `online` status.

The `fschstate` command allows the user to report or to change the state of a drive component or storage subsystem.

Note

For more information on the `fschstate` command, refer to the *Command Reference* book.

The `fschstate` command can be executed when FileServ is active or nonactive. Only storage subsystems can be changed if FileServ or VolServ is inactive.

Drive component changes require both FileServ and VolServ to be active. Valid states are:

- UNAVAIL
- MAINT
- ON
- OFF
- UNKNOWN

Because FileServ and VolServ components work interactively, changes in the drive components or storage subsystem are reflected in VolServ.

If a component is taken to either the `offline` or `maintenance` state, FileServ does not attempt any processing with that component.

When a component is taken to a `maintenance` state, the maintenance port is enabled and the Ethernet communications link is disabled; whereas, the `offline state` is only a logical state within FileServ.

After maintenance has been completed, change the component state back to the `online` state for the component to be reused by FileServ.

Using the `fschstate` command without any options generates a report that shows all currently configured library components, for example, drives, drive identifiers, and VolServ systems. The report can be limited to a single component by specifying a component alias.

Step 1. Run the `fschstate` command and specify the component alias of any storage subsystem, component or drive in the library system.

Step 2. Use the `-s` option to indicate the required new state of the component. Valid state entries are: `UNAVAIL`, `MAINT`, `ON`, `OFF`, and `UNKNOWN`.

Step 3. To receive a report on all configured storage subsystem and drive components in the system, run the `fschstate` command with out options.

Add and Delete Drives

Adding and deleting drives from an Automated Media Library (AML) requires technical support assistance. The robot inside the AML requires reteaching and the robotic database will need to be update with special support equipment used by trained ADIC technical personnel.

After this is done, the drives can be configured by the system administrator.

To add a drive, follow the steps below:

Step 1. Performed by ADIC technical support.

- Install drive in AML configuration.
- Teach robot (teaching automatically updates the robotic database).
- Delete library.
- Create new library.
- Remap the library.

Step 2. Performed by VolServ system administrator.

- Define new tape drive and specify media type.
- Associate new drive with VolServ library being used by FileServ.

Step 3. Log in as `root` user.

Step 4. Change directory to `$FS_HOME` (directory where FileServ is located).

Step 5. Source FileServ `cshrc` by entering `source .cshrc`.

Step 6. Execute the `fsvsinstall` script by entering `run /util/install/fsvsinstall` and respond to questions.

The `fsvsinstall` script prompts you for the following information:

- Enter VolServ library used by FileServ software.
- Enter drive identifier to add to FileServ.
- Enter component alias, device pathname, drive type, and drive delay time for each drive identifier.

Step 7. Log out as `root` and log in as the FileServ system administrator.

Step 8. Verify the drives have been added to the FileServ system by entering `fsconfig`.

Step 9. Log out as the FileServ system administrator and log in as the VolServ system administrator.

Step 10. Verify the drives have been added to VolServ by querying the drive pool used by FileServ.

Step 11. Log out as the VolServ system administrator.

Fine-tune Parameters

The following parameters can be fine-tuned for better system performance:

Reallocate Drives

Drive allocation is affected by the following sets parameters:

- **Drive deallocation**.
  The deallocation parameters determine when a drive is taken `offline` because of access errors. Drive deallocation parameters are set by modifying the appropriate system parameters. The system parameter file, `fs_sysparm`, is located in the `$FS_HOME/sysparms` directory.

- **Delayed dismount**.
  The delayed dismount parameters determine when media is dismounted from a drive after a store or retrieve request is processed. The delayed dismount feature allows optimization of reallocation of media for another request in the queue and, thus, can reduce mount/dismount time for resource requests.

Step 1. Using a text editor, open the `fs_sysparm` file located in the `$FS_HOME/sysparms` directory.

Step 2. Modify any of the drive parameters identified in the table below:

Parameters	Description
FS_DRIVE_ERR_THRESHOLD	Threshold value when equalled or exceeded results in drives being taken offline.
FS_THRESHOLD_INC_NUM	Threshold increment number variable. Value for the drive failure level when an access to a drive fails.
FS_THRESHOLD_DEC_NUM	Threshold decrement number variable. The value for the drive failure level when access to a drive is successful.

Step 3. Save and close the edited file.

Modify Tape Usage

The way data is written to tape is modified by editing the appropriate system parameters that control:

- Estimated tape length calculations.
- Data block size.

Step 1. Using a text editor, open the `fs_sysparm` file located in the `$FS_HOME/sysparms` directory.

Step 2. Modify any of the default tape utilization parameters identified in the table below:

Parameters	Description
NOMINAL_FILE_SIZE	Nominal file size in bytes. Used to estimate remaining tape capacity.

Parameters	Description
DEF_MED_SPC_3480	Default 3480 tape length in bytes.
DEF_MED_SPC_3490	Default 3490 tape length in bytes.
DEF_MED_SPC_8590	Default 8590 tape length in bytes.
DEF_MED_SPC_CTIII	Default CTIII tape length in bytes
DEF_MED_SPC_CTIV	Default CTIV tape length in bytes.
DEF_MED_SPC_D2SM	Default small D-2 tape length in bytes.
DEF_MED_SPC_D2MD	Default media D-2 tape length in bytes.
DEF_MED_SPC_D2LG	Default large D-2 tape length in bytes.
FS_DISK_BLOCK_FACTOR	Number of disk blocks to use per transfer when reading/writing the disk (default = 10).
FS_TAPE_BLOCK_FACTOR	Number of disk blocks to use per transfer when reading/writing a tape (default = 1).
FS_RSP2150_BLOCK_FACTOR	Number of 2150 blocks to use per transfer when reading/writing the 2150 (default = 4).

Step 3. Save and close the edited file.

Redirect Policy Applications

Step 1. Open the `crontab` file.

Step 2. Modifying the list of DataClass groups where the policies is applied.

NOTES

2

Operating Tasks

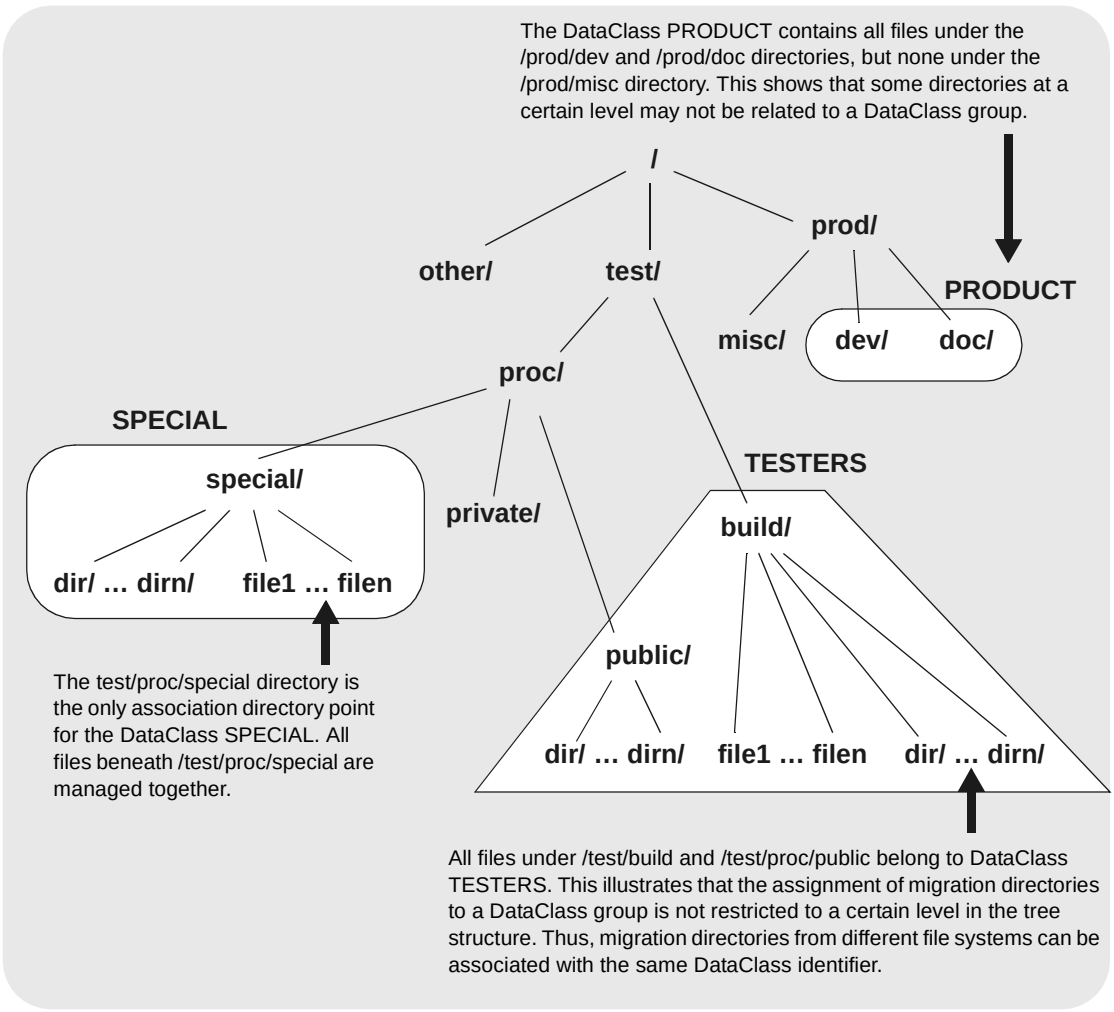
Roadmap

Topic	Refer To Chapter
Initialize FileServ, configure interface with VolServ, perform start up and shut down, modify system configuration, and tune system for better performance.	1
Manage: DataClass Groups, files, migration, data, and media.	2
Using the FileServ utilities.	3
Troubleshoot operating problems.	4

DataClass Management

Files controlled by FileServ are segregated into DataClass groups. A DataClass group contains a unique set of parameters that govern the migration behavior of the directories that are associated with the class.

The figure below depicts directory-to-DataClass relationships:



Migration policy parameters are configurable on a DataClass group basis, which provides the FileServ administrator with control over the behavior of the FileServ system. The minimum time required on disk before migration and file cleanup action are examples of DataClass migration policy parameters. Quotas for media usage (both a soft warning limit and a hard limit) are also kept on a DataClass group basis. DataClass group parameters are recorded so that tuning is accomplished while the FileServ system is active.

FileServ uses DataClass groups as the means to segregate files on media. All files on media are of the same DataClass group. This provides a level of physical security for those sites for which it is required. This segregation ensures that retrieval of files from different user groups (as defined by the DataClass group parameters) does not require access of the same physical media.

DataClass Groups

Characteristics of DataClass groups are established and modified so that data is handled differently among DataClass groups. Initially, configuring or changing the parameter settings for DataClass groups involves DataClass manipulation. This allows flexible handling of data to suit the user's needs.The table below lists the tasks performed on DataClass groups:

Note

All GUI class operations are run from **ADMIN-> Define Class ->Create Groups** or with the `fsaddclass` command.

To create a DataClass group, follow the steps below.

Tip

A DataClass group must be defined before directories can be associated with it.

Step 1. To verify the parameter settings before attempting any modification, use the `fsclassinfo` command to view the DataClass group parameters.

Step 2. Run the `fsaddclass` command and specify a DataClass group name.

```
% fsaddclass prodclass
```

Step 3. Use other appropriate options, as needed. For option information, refer to the *Command Reference* book.

Modify Group with fsmodclass Command

The `fsmodclass` command modifies DataClass group parameters.

Note

For more information on the `fsmodclass` command, refer to the *Command Reference* book.

Step 1. Run the `fsmodclass` command and specify a DataClass group name.

```
% fsmodclass prodclass -n joe
```

Step 2. Use `fsclassinfo` to compare each DataClass group parameter. Then either change the class parameters for the new class to allow spanning or change the class parameters for the old class to disallow spanning.

- Specify *oldclass.*
- Specify *newclass* (*newclass* name cannot be the same as *oldclass* name). The directories associated with oldclass become associated with newclass, and oldclass is deleted.

```
% fsclassrnm prodclass iradclass
```

Rename Group with fsclassrnm Command

The `fsclassrnm` operation allows the name of a DataClass group to be changed.

Note

For more information on the `fsclassrnm` command, refer to the *Command Reference* book.

All files and media associated with the previous DataClass group still exist in the new DataClass group. If the new specified DataClass group already exists, the previous DataClass group is merged into the new DataClass group. No restrictions exist when merging two DataClass groups. If two DataClass groups use different media types, the merge still occurs.

Remove Group with fsrmclass Command

The `fsrmclass` operation removes a DataClass group and its parameters from FileServ.

Note

For more information on the `fsrmclass` command, refer to the *Command Reference* book.

This procedure also deletes the directory-to-DataClass associations that involve the specified DataClass group. Empty all directories in the DataClass group before the `fsrmclass` operation is executed. If any association point directory is not empty, the directory is not deleted and remains in the file system unassociated with any DataClass group. A status message is displayed that indicates the directories are, or are not successfully removed. The UNIX command, `rmdir`, is used to delete the directory from the UNIX system.

Generate Reports with fsclassinfo Command

The Class Information Report displays DataClass groups and parameters.

For the specified DataClass groups, the `fsclassinfo` command or `Class Information` screen lists the parameters and the points of each DataClass association.

Note

For more information on the `fsclassinfo` command, refer to the *Command Reference* book.

The long format of the report shows a listing of directory-to-DataClass association points. The long option gives the DataClass group and its parameters, followed by a list of directories associated with each DataClass group.

The short report is the default and lists each specified DataClass group and all its parameters.

If no class is specified, a short report on all DataClass groups is generated.

The generated report is sent to `stdout` but can be redirected to a file or piped to a printer.

The output of a long report is shown below:

```
% fsclassinfo low_twr6 -l
-----------------------------------------------------------------
 Class Information Report Sun Jan 31 14:18:03 1999
 Class ID: low_twr6
-----------------------------------------------------------------

    Soft Limit:   80             Max Copies:   2
    Hard Limit:   95             Media Type:   D2S
     Notify ID:   adic         File Spanning:   n
 Security Code:   1CPY          File Cleanup:   IMMEDIATE
Account Number:   PTRU         Media Cleanup:   SYSTEM
    Drive Pool:   fspool     File Clustering:   n
Default Copies:   1            Mintime (min):   1

Associated Directories:
 /arch1/low_twr6
 /arch3/XLfiles/low_twr6

FS0000 31 174105 fsclassinfo completed: Command Successful
```

The table below describes the report's fields:

Field	Description
Soft Limit	Number of media used for the DataClass group.
Hard Limit	Number of media allocated for the DataClass group.

Field	Description
Notify ID	E-mail address where notification is sent when the soft limit or hard limit is reached.
Security Code	Four-character DataClass group security code.
Account Number	Five-character DataClass group account number.
Drive Pool	Name of drive pool associated with the DataClass group.
Default Copies	Number of copies stored on media. **Note**: The `Default Copies` parameter defines the required number of copies that must be made.
Max Copies	Maximum number of copies allowed for each file associated with the DataClass group.
Media Type	Media type for the DataClass group.
File Spanning	Indicates whether file spanning of media is enabled for the DataClass group.
File Cleanup	File cleanup action for the DataClass group.
Media Cleanup	Media cleanup action for the DataClass group.
File Clustering	Indicates whether file clustering is enabled or not enabled.
Mintime	Minimum time (in minutes) a file must reside on disk before being eligible for storage, or for removal from disk if all file copies reside on media.

Manipulate Relations

Migration directories are defined by associating directories with a DataClass group. A DataClass group is associated with one or more directory paths. This association is defined as a relation between the DataClass group and the directory. The directory is known as the association point. Relations are added and deleted in a file system.

The table below lists the tasks performed on DataClass groups:

Note

All GUI relation operations are executed from **ADMIN->Define Class/Directory Relations**.

Add Relations with fsaddrelation Command

A relation is added with the `fsaddrelation` operation.

Note

For more information on the `fsaddrelation` command, refer to the *Command Reference* book.

This procedure allows you to associate a DataClass group definition with a directory. All files and directories beneath the association point are included in and governed by the DataClass group parameters.

To add a DataClass group association to a directory, it must not be superior to or subordinate to any directory that already has a Directory-to-DataClass relationship.

If the subdirectories already have the same DataClass relationship, the association point can be rolled up. For more information, see "Change Relation Points" on page 2-17. The DataClass association point cannot be a directory in the `root` file system unless it is a mount point for a new file system.

The DataClass group associated with a directory also determines the media to which the files in that directory is migrated. The files in a migration directory are only migrated to media that contain files with the same DataClass group association.

Associate New Data with New Relations

After adding a new relation, run the `fsdump` command to associate any new data with the new relation (the new mount point).

Caution

Failure to run `fsdump` after adding relations may impact system performance.

The `fsdump` command creates an `fs_dump.file` that contains an entry for each associated directory and stored file in the file system. The `fsaudit` command uses the `fs_dump.file` to reconcile the restored files from the dump tape and the FileServ database.

Associate New Directories with Files

Files can be present in a directory when the `fsaddrelation` is performed.

Creating a directory under a directory that is an association point automatically associates the newly created directory with the DataClass group of its parent directory.

Note

When adding a DataClass relation to a directory that contains files, no users are allowed in the file system at the time the command is issued or the command fails.

Returned status shows that the file system is busy. A file system must be inactive because it must be unmounted, mapped, and remounted at the time of the fsaddrelation to ensure that associated file information is accurate upon completion.

The file system is mapped by unmounting the file system and remounting it exclusively for the mapping process. It is recommended to perform the `fsaddrelation` to a populated directory during a slack time. Users cannot access any files in the file system during the fsaddrelation processing.

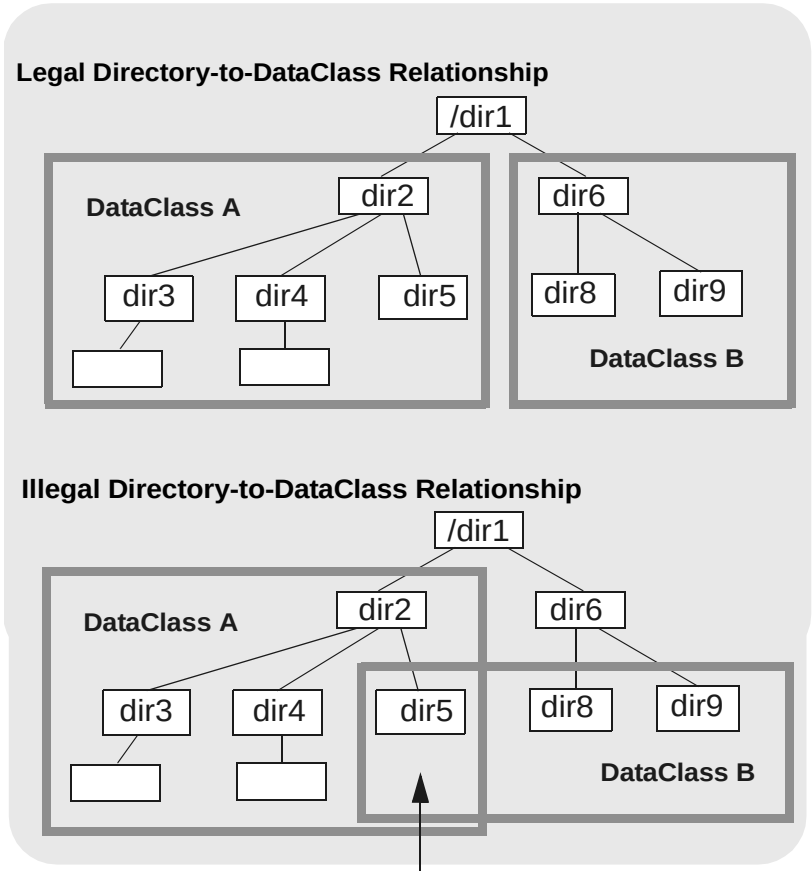
Following the mapping and execution of the `fsaddrelation` operation, the file system is unmounted and remounted to allow user access again.

Because the `fstat` options are limited, list all open files and `grep` the major and minor device numbers of the file system to check for users. On SGI systems, the user can run the `fuser` command to find the same information.

Note

For more information about major and minor device numbers, refer to the manual page for the UNIX `fstat` or `fuser` commands.

Major and minor device numbers are not necessary. Instead, use the device name of the file system as listed by the UNIX `df` command.



Below is an example of the `df` command. In the example, the numbers given by **fuser** are processes accessing files in `/tst1`. To get more information on the process, use `ps -elf` and `grep` for the process number.

```
% df

Filesystem        Type  blocks     use   avail  %use  Mounted on
/dev/root          efs  205614   46958  158656   23%  /
/dev/usr           efs 2068050 1678616  389434   81%  /usr
/dev/dsk/dks1d1s2  efs 5680906 4089070 1591836   72%  /fs
/dev/dsk/dks3d3s0  xfs 1949616 1103528  846088   57%  /lst1

%fuser/dev/dsk/dks3d3s0
/dev/dsk/dks3d3s010620co 11762c 10738co
```

Example

Step 1. Use the UNIX **ls** command to verify that the directory to be associated already exists.

```
% ls
data
irad
```

Step 2. If the directory does not exist, use the UNIX `mkdir` command to create the directory. The specified directory cannot be a directory in the root file system unless it is a mount point for a new file system.

```
% mkdir production
% ls
data
irad
production
```

Step 3. Verify that the DataClass group is defined by using the `fsclassinfo` command.

> **Note**
>
> The `fsclassinfo` command displays information for all DataClass groups.

Step 4. If the DataClass group is not defined, see "All GUI class operations are run from ADMIN-> Define Class ->Create Groups or with the `fsaddclass` command." on page 2-5.

Step 5. Run the `fsaddrelation` command and specify the DataClass group and directory to be associated.

```
% fsaddrelation production -c england
```

> **Note**
>
> The entire path name need not be entered. The directory path is resolved using the current working directory. The directory notation, "." (current directory) and "." (previous directory), can be used.

Step 6. After adding a relation to a directory, the directory's file system is dumped so that the parent directory of the relation point is present in the dump file. If several relations are added to the file system, the file system dump is performed after all the relations are made.

```
% fsdump -f /arch1/site1_production.dumpfile /site1/production
```

Remove Relations with fsrmrelation Command

A directory-to-DataClass relationship is removed and the directory is made a nonmigration directory by using the `fsrmrelation` operation.

> **Note**
>
> For more information on the `fsrmrelation` command, refer to the *Command Reference* book.

Removing a DataClass relationship from a directory causes it to be a nonmigration directory by FileServ. Therefore, files that are copied, saved, or moved into this directory by users (after the directory is disassociated from a DataClass group) are not migrated to media. The relationship is restored, and the directory is made migration by using the `fsrmrelation` operation.

Before issuing the `fsrmrelation` operation, all files and subdirectories in the association point directory must be deleted or moved so that the directory is empty. Directories are not deleted by `fsrmrelation`.

Step 1. Run the `fsrmrelation` command.

> **Note**
>
> The directory must be empty before the `fsrmrelation` command is allowed. If the directory contains files or other subdirectories, the `fsrmrelation` command fails.

Specify the name of the directory to disassociate.

```
% fsrmrelation production
```

Change Relation Points

Association points are moved up or down in the file system hierarchy tree from their original definition with the `fsrollup` and `fsrolldown` commands, or `Define Class/Directory Relations` menu.

This is a useful operation in adjusting the size and manageability of migration directory structures associated with DataClass identifiers.

Roll Up DataClass Groups

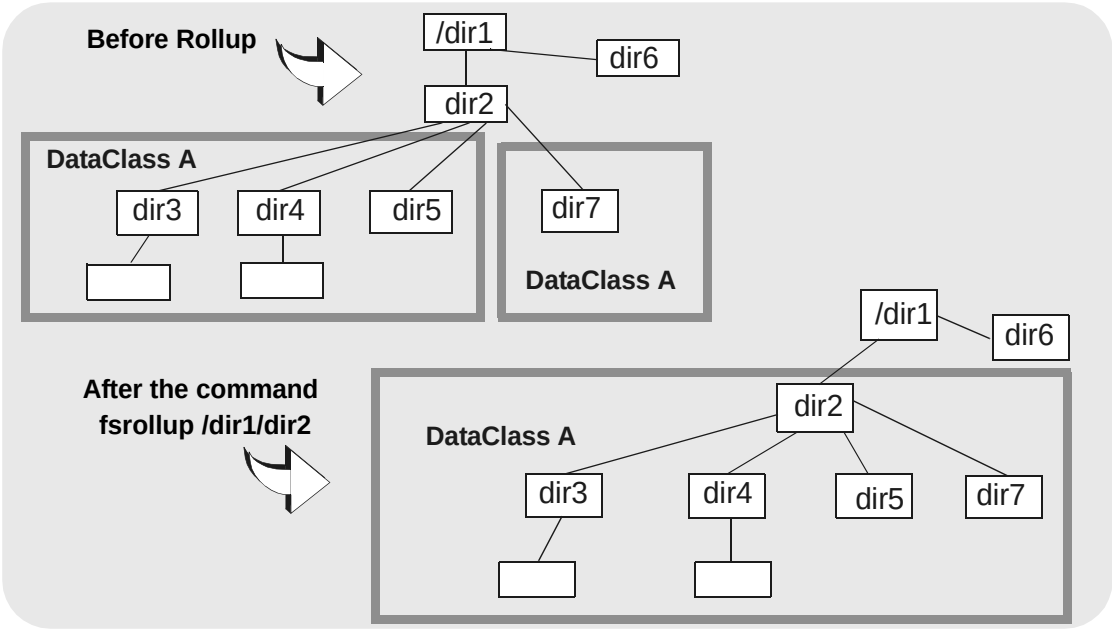
The common parent directory of one or more association points in a DataClass group is included in the DataClass group using the `fsrollup` command.

Note

For more information on the `fsrollup` command, refer to the *Command Reference* book.

This causes FileServ to associate the specified directory in the command with the DataClass group of its subordinate directories. The entries in FileServ that associate the subordinate directories with the DataClass group are removed. The specified directory becomes an association point in the DataClass group. If the specified directory includes either subordinate directories associated with a different DataClass group, or nonmigration directories, the `fsrollup` command fails.

In the figure below, the `fsrollup` command fails to roll up to `dir1` from `dir2` because `dir6` is not a migration directory.



The `fsrollup` function is performed on only one hierarchical directory level at a time. Roll up of a DataClass group to a directory that is two or more levels above an upper boundary directory of the DataClass group is not allowed. The user cannot roll up past a file system mount point. The specified directory contains only subordinate directories and no other files.

Perform this adjustment before many files are stored to the directories.

Step 1. Verify that the rollup does not cause contention with other migration directories

Step 2. Run the `fsrollup` command.

Step 3. Specify the name of the directory to be rolled up to (the new association point).

```
% fsrollup production
```

Note

All subordinate directories of *directory* must be in the same DataClass group. The directory specified in *directory* can only contain subordinate directories and no other files

Roll Down DataClass Groups

The DataClass association point directory level is changed using the `fsrolldown` command.

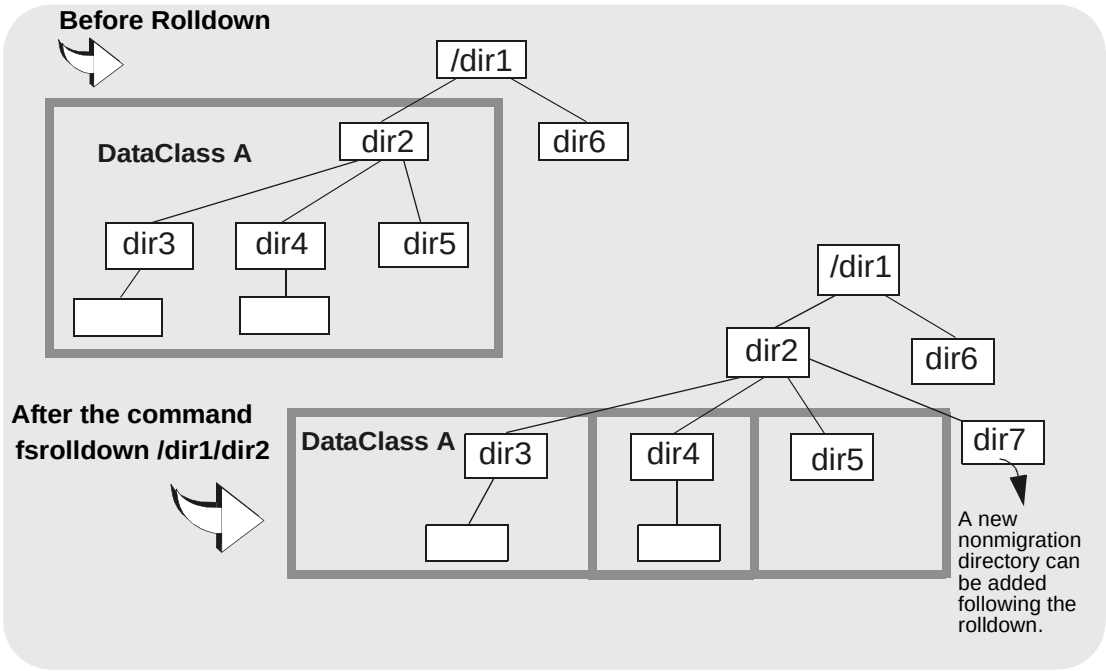
Note

For more information on the `fsrolldown` command, refer to the *Command Reference* book.

A directory at the association point of the DataClass group is excluded from the DataClass group by specifying it in the `fsrolldown` command. This causes each subdirectory of that directory to become an association point for the DataClass group. The specified directory no longer maintains a directory-to-DataClass relationship and, therefore, is a nonmigration directory. The specified directory contains only subordinate directories and no other files. When rolling down a directory, files must be moved into directories before the command is invoked. It is preferable to perform this adjustment before many files are stored to the directories.

The association is only rolled down to an association point for each directory under the original association point. The Roll Down operation is performed on only one hierarchical directory level at a time.

FileServ does not permit you to roll down to a directory that is two or more levels below an association point of the DataClass group.



Step 1. Ensure that no regular files are in the specified directory. The only nodes are subdirectories, as these each become new association points.

Step 2. Run the `fsrolldown` command. Specify the name of the directory to roll down (the old association point).

```
% fsrolldown production
```

Note

The directory specified in *directory* can only contain subordinate directories and no other files.

Generate DataClass Report

While DataClass relation points are listed in the long format of the Class Information Report, individual directories at any position in the file system can be checked for their DataClass relationship with the `fsdirclass` command or `Directory to Class Information` screen.

Note

For more information on the `fsdirclass` command, refer to the *Command Reference* book.

The Relation Information Report displays the class associated with the specified directory. If the specified directory is not associated with a class, FileServ returns an error message.

Step 1. Run the `fsdirclass` command. Specify the name of the directory to be examined.

Step 2. An example of the output is shown below:

```
% fsdirclass /production

FS0070 02 03881 fsdirclass completed:
/production located in class england.
```

File Management

File management tracks file information, requests file copies, and controls disk space of files under FileServ control.

The table below lists file topics:

Pathname

The directories and subdirectories under a migration point must have a full path name less than or equal to 255 characters.

Note

If a directory name is created with greater than 255 characters, the system administrator receives a caution on the console and logfile to immediately remove the directory. Files in this directory cannot be stored.

Links

FileServ supports hard links and soft links.

Note

A hard link can be created after a file is stored or the `fschfiat` command is performed on the file.

When hard-linked files are stored and retrieved, the common file data is stored or retrieved for all of the links, no matter which file name is submitted with the `fsstore` command.

The return status for the operation lists a file name of any one of the links and not necessarily the file name submitted as an argument with the command. If a different linked file name is returned in the status, the file data was correctly handled.

FileServ prevents hard links between files across DataClass boundaries.

Symbolic links can cross DataClass boundaries and file system boundaries.

Tip
ADIC does not recommend you use symbolic links across DataClass groups.

The DataClass parameters that control the original file are the DataClass groups of the file links, regardless of the location. Soft links that cross DataClass groups are misleading, because the file data can actually reside in a different class.

Spanning

FileServ supports file spanning across media. If a file is larger than a single piece if media, the file will *span* to another piece of media until the entire file is stored.

A file that spans multiple media always starts at the beginning of each additional media required to store the file. The file is stored on media until all available space is used, the remaining part of the file (spanned portion) is stored on a blank media. If the file overflows another media, another blank is used until the entire file is stored.

File spanning is enabled or disabled for each DataClass group by the `fsaddclass` command. The `fsmodclass` allows modification of DataClass group processing parameters while the `fsfileinfo` command reports all the locations of the spanned file.

Clustering

FileServ supports file clustering. File clustering is the accumulation of files associated with a DataClass group. When these files are stored on media, they are seen as one logical file. Any file in a cluster is retrieved as any other non-clustered FileServ file.

File clustering can be defined when adding a DataClass group to the file system (`fsaddclass`) or by the `fsmodclass` command. File clustering offers two advantages over individual file storage. The first advantage is the media space saved when clustering. The amount of space consumed by a cluster of files is less than the space used if files are stored individually. The second advantage is that the amount of time to store a cluster of files is less than the sum of the time required to store each file separately in the cluster. Clustered file information is sent in one continuous stream of data to the drive for storage on media. Unclustered files require the drive to stop, insert tape marks, file labels, and perform database updates for each file, which results in increased storage time.

The system administrator controls the maximum size of the files to be clustered and the clustered file size for all supported media types.

The *c*`cluster.config` file, located under the `FS_HOME`/`sysparms` directory, controls file clustering parameters. Each DataClass group is added to this file and the applicable file size values are entered (file sizes are expressed in millions of bytes). If the DataClass groups are not added, the `filesize.config` default values defined in the file are used. The file is read each time a list of files is marked for storage. This file is dynamic and enables the system administrator to make changes without recycling FileServ.

```
##############################################################################
# Copyright 1999 ADIC, Inc.
##############################################################################

#
#This is a FileServ file for clustering configuration
#control to be applied by the store policy code
#
#Any line that starts with a "#" is considered a comment
#

#Each entry should have the following, in order, and blank separated:
#
#- Class name              up to 16 characters)
#- Max file size to cluster(any file larger than this will NOT
                           be part of a cluster)
#- D2 cluster size         (largest clustered file size for D2)
#- 3xxx cluster size       (largest clustered file size for 3490,
                           3490E,3480,and 8590 media types)
#- Metrum cluster size     (largest clustered file size for Metrum)
#- DLT cluster size         (largest clustered file size for DLT media
#                          types which include CTIII and CTIV
# All the above sizes are in millions of bytes
#
```

```
#########################################################################
#The following is a sample line which has been commented out
#myclass          30 1000  30   50  150
#
#########################################################################
#
# Never ever change the entry for default unless specifically
# directed by technical support personnel.
#########################################################################

#
# $LOG$
#

#########################################################################
# $Id: cluster.config,v 5.1.0.2 1999/06/21 19:41:14 johnb Exp $ $ADIC
*
#########################################################################
<default>          20   1000    30    50    150
```

Modify File Attributes

Files that reside in directories associated with a DataClass group have two types of attributes:

- Attributes that specify the number of copies of the file to store.
- Attributes that designate how policy is applied.

The default setting for these attributes are acquired from the DataClass group defaults set by the system administrator when the DataClass group is defined.

DataClass group attributes are modified on a file-by-file basis using the `fschfiat` command.

Note

For more information on the `fschfiat` command, refer to the *Command Reference* book.

Files can then share DataClass group parameters with other files in the DataClass group. These files deviate in how policy is applied and in the number of copies of that file that are allowed on media. The `fschfiat` command allows fine tuning of the file's attributes without changing the characteristics of the entire DataClass group.

For example, the common home directory files *.cshrc* and *.login*, are used each time a user logs into the system. However, they are excluded from the cleanup policy application to avoid a delay during the login process.

For example, the `fsstore -f i` (truncate immediately after storing) or `-f p` (truncate according to the policy application) options cannot be used if the file is excluded from truncation.

Step 1. Run the `fschfiat` command.

Use the appropriate options, as needed. For option information, refer to the *Command Reference* book.

Step 2. Verify the change by running the fsfileinfo command.

```
% fschfiat -i test1
```

Generate Report

To view information about file location and attributes, follow the steps below:

Step 1. Run the `fsfileinfo` command, specifying the names of the files you want reports on.

Step 2. For files stored on media, this report returns an identifier for each media where the file is located. Each media ID is accompanied by a number that represents which copy of the file is on the media. An example of the output is shown below:

```
% fsfileinfo /arch1/heavy_twr1/small_010316_1/small_1

------------------------------------------------------------
 File Information Report Wed Feb 3 14:10:12 1999
 Filename: /arch1/heavy_twr1/small_010316_1/small_1
------------------------------------------------------------

Creation time:3-Jan-94 16:09:40
        Owner:   testa              File size:   776839
        Group:   mss                 Location:   TAPE
       Access:   664          Existing Copies:   2
        Class:   heavy_twr1     Target Copies:   2
        Trunc:   IMMEDIATE

Media:   TST0917(1)  TST0909(2)     (1 = primary, 2 = secondary)

FS0000 03 203981 fsfileinfo completed: Command Successful.
```

Field	Description
Owner	
Group	DataClass group governing the file.
Access	

Field	Description
Class	
Trunc	Identifies whether the file is a candidate for cleanup policy application.
File size	Size in bytes.
Location	Identifies whether file resides on disk, tape, or both.
Existing Copies	Number of copies of the file that currently exist.
Target Copies	

Data Migration

A migration directory is controlled by FileServ (for example, a directory that has a DataClass association.) Files stored under this directory, as well as any subsequent subdirectories, are migrated from disk to media as specified by the DataClass parameters.

A DataClass group can be created so that files of associated directories exhibit certain distinctive migration behavior. These behaviors are controlled by a combination of setting the:

- File Cleanup.
- Minimum Time parameters.
- Policies.

These are described below.

File Cleanup

After a file is copied to media (migrated), the disk copy of the file remains or is removed from the disk. The `fschfiat-e` command allows a file to remain on the disk indefinitely.

The file retention for the DataClass group definition defines how a disk copy is removed. By setting this parameter to *immediate*, after a file is known to contain the required number of copies on media, the disk copy is immediately removed. By setting the parameter to "`policy application`", the file remains on disk even after the required number of media copies exist.

The amount of time the file remains on disk after storage is partially determined by setting the DataClass Minimum Time Parameter *(*`mintime`*)*. As long as the file system contains empty disk space (as set by the LOBLK_THRESHOLD in the `fs_sysparm` file on SGI systems), the file remains on disk until the Minimum Time interval passes.

Minimum Time Parameters

The `mintime` is the minimum amount of time after the file is written or read before any policy action is taken. The mintime determines eligibility for file storage and for file removal from disk. For the cleanup policy, `mintime` is calculated as the time elapsed after a file was stored on media before being removed from disk.

Setting both the `File Cleanup Action` and `Minimum Time` parameters tend to keep available disk space high by freeing disk blocks as soon as possible (the *immediate* setting). These parameters keep active files available on disk as long as possible, thus freeing disk space as files become inactive.

Note

In this case, active means frequently accessed.

However, setting the `File Cleanup Action` and `Minimum Time` parameters does not establish the DataClass migration behavior of the files.

Rather, the planned execution of migration and disk cleanup policies as well as the DataClass characteristics provide the overall migration behavior for DataClass groups.

Policies

Migration management includes the application of storage and cleanup policies using the `fspolicy` command with various options. Changing the overflow parameters in the `fstab` file also affects migration.

Migration and disk cleanup policies are implemented in the following policies:

Storage Policy

Storage policy is used to migrate eligible file data to media. When the policy is initiated, a list is generated to identify the files that must be stored to media. The list contains files that do not have a current set of media copies and are not accessed in the time specified by that DataClass group's `mintime` parameter.

Storage policy is performed for a DataClass group using the `fspolicy -s` command. For more information, see "Using the fspolicy Command" on page 2-35

Note

To run `fspolicy` to store a very large number of files, (a million or more) make sure you have sufficient memory. The required memory should equal the maximum number of files to store times 425.

File storage and media type selection are based upon the file size controlled by the system administrator. By entering the DataClass group name and defining file sizes with the applicable media type in the `filesize.config` file (located under the `$FS_HOME/sysparms` directory), the system administrator customizes file storage policy for each DataClass group.

The `filesize.config` file as shown in the example below, is read each time a list of files is received for storage. This file is dynamic and enables the system administrator to make changes without recycling FileServ.

```
#########################################################
#This is a FileServ file for placing files on different
#media types based on the size of the files themselves.
#
#Any line that starts with a "#" is considered a comment
#

#Each entry should have the following, in order, and blank
separated:
# - Class name (up to 16 characters)
# - Copy Number (1-2, *: all copies))
# - file size range and media type
# - Valid mediatypes: D2S, D2L, D2M, 3490, 3490E, and 3480
#
#All the above sizes are in millions of bytes
# * - Indicates all other unspecified sizes.
#

# NOTE: Specified ranges may intersect one another however the
# mediatype is determined by parsing the text line from left to
right.

#########################################################
#The following is a sample line which has been commented out
#myclass    1  0-30:3480     30-1000:         25-*:D2S
#myclass    2  0-50:3480     50-1000:         20-*:D2S
#myclass1   *  0-25:         30-1000:D2S      25-2500:D2M *-*:D2L
#myclass2   *  30-1000:D2S
#########################################################
#The default is the mediatype specified in the class.
#########################################################

doc_test   1  0-25:D2S      30-1000:3480          Example only
```

Cleanup Policy

Cleanup policy is used to remove file data from disk. The number of files removed is based on the available disk space requested for a file system. When the cleanup policy is run, each file that is a disk data truncation candidate is assigned a weight. This weight is a factor in determining to remove one particular disk file over another.

A disk file becomes a candidate for cleanup if it exhibits the following characteristics:

- The required number of copies are stored to media
- The file attributes are not set for exclusion from cleanup processing
- The file is idle on disk for its *mintime* interval

When cleanup policy is applied, file data is truncated beginning with files of the highest weight and continuing down the list. Cleanup policy application stops after the file system reaches its specified free space percentage or when the list is exhausted. If the list is exhausted before the specified free space is reached, the returned status indicates this as a failure to reach the required percentage fill level.

To obtain more free space, additional files must be made eligible for truncation and the cleanup policy must be rerun. Files are made available by reducing relevant *mintime* values (either on a DataClass group or on specific files), or by ignoring the *mintime* check by executing the policy with a *force* option. Generating `fsstore` commands for disk-only files also makes additional files available for disk truncation. The cleanup policy is initiated on a file-system basis with the `fspolicy -t` command.

Overflow Policy

The `HIBLK_THRESHOLD` and `LOBLK_THRESHOLD` system variables are located in the `fs_sysparm` file. These variables are system-wide parameters.

FileServ reads the new values at startup time. If FileServ is running, you must recycle the system to pickup the new values.

If the overflow `hiblk` value is reached for a file system, the `$FS_HOME/internal/policy_dir/fs_spt_ovflow` script is automatically invoked. This script is used for:

- Emergency situations when routine data cleanup is not maintained for a file system,
- Or, when data is rapidly stored to the disk before the routine storage policy reacts.

> **Note**
>
> **SGI ONLY:** If `HIBLK_THRESHOLD` and `LOBLK_THRESHOLD` parameters are not given to a file system, the emergency script is never invoked for that file system, regardless of the disk usage.

- Using the fspolicy Command

The `fspolicy` command is run to store eligible files from disk to media or to truncate files on disk that are fully stored to media

> **Note**
>
> For more information on the fspolicy command, refer to the *Command Reference* book.

A store request is processed in two ways:

- Normal. The normal store request is placed in the queue for processing along with all other resource requests.

> **Note**
>
> To run `fspolicy` to store a very large number of files, (a million or more) make sure you have sufficient memory. The required memory should equal the maximum number of files to store times 425.

- Emergency.

The `fspolicy` command maintains an internal store and truncate candidate list of files. When the command is invoked, those candidates meeting the criteria for storage or truncation are processed. This candidate list can be overridden with the `-w` option that scans the entire file system looking for potential store and truncation candidates that meet the defined DataClass group *mintime* requirements. Although this process is database intensive, it should be performed on a regular basis as well as any time an interrupt occurs.

This option should be run during times of minimal FileServ software activity or executed through `cron`.

Running as a cron Event

To invoke `fspolicy` on a periodic basis, use the UNIX `cron` utility.

> **Note**
>
> For more information on `crontab`, refer to the man pages.

The `cron` utility is a permanent process that is started by `/etc/rc.local`. This utility consults the files in the `$FS_HOME/.crontab` directory to determine which tasks to perform and at what time.

Each line in a `.crontab` file must consist of six positional fields, separated by spaces or tabs, and formatted as follows:

```
minutes  hours  day-of-month  month  day-of-week   command
```

Any of the first five fields can be a list of values separated by commas. A value is either a number, or a pair of numbers that are separated by a hyphen. This value indicates that the job is performed for all the times in the specified range. If a field is an asterisk character (*), the job is performed for all possible values of the field.

Field	Description
minutes	Minutes field. Values range from 0 through 59.
hours	Hours field. Values range from 0 through 23.
day-of-month	Day of the month. Values range from 1 through 31.
month	Month of the year. Values range from 1 through 12.
day-of-week	Day of the week. Values range from 0 through 6. Sunday is day 0. For backward compatibility with older systems, Sunday can also be specified as day 7.
command	Command to be run. A percent character in this field is translated to a NEWLINE character. Only the first line (up to a per cent or end of line) of the command field is executed by the shell. The other lines are made available to the command as standard input.

The specification of days is made by two fields (day of the month and day of the week). If both are specified as a list of elements, the policy adheres to both. If one of the fields is an asterisk, the other field determines the day of the month or week.

The example below shows this entry runs a command at midnight on the 1st and 15th of each month, as well as every Monday.

```
0 0 1,15 * 1
```

To specify days by only one field, the other field is set to *.

```
0 0 * * 1
```

When used to run `fspolicy` as `root`, the `cron` utility is placed under the `FileServ` user ID home directory. Any generated output or errors are mailed to files location `root` or `FileServ` user unless they are redirected.

The example below shows `fspolicy` running periodically on several different DataClass groups at different times in the following example of the `.crontab` file.

```
0 0 * * * /usr/adic/exec/fspolicy -s -c prodclass
15 0 * * * /usr/adic/exec/fspolicy -t -y /dev
15 4 * * * /usr/adic/exec/fspolicy -s -y groupaclass
40 4 * * * /usr/adic/exec/fspolicy -s -y groupbclass
0 0 * * 1-5 /usr/adic/exec/fspolicy -s -y tempclass
0 0 * * 0,6 /usr/adic/exec/fspolicy -s -y permclass
```

- A storage policy for `prodclass` runs at minute 0 of hour 0 (midnight) of every day.
- The cleanup policy is invoked for the file system */dev* at 15 minutes after midnight every day.

- The two cases where storage policy is run on `groupaclass` and `groupbclass` run at 15 minutes past 4 and at 40 minutes past 4, respectively, every day of the year.
- The storage policy application for `tempclass` is run at midnight on weekdays. Finally, storage policy is run for `permclass` at midnight on weekends only.

Using this method, policies are run at whatever intervals are required. However, use caution to ensure that the interval between executions of a particular policy is set so that more than one instance of the policy is not initiated simultaneously. System resources are wasted if the policies are duplicated by overlapping their execution.

After the `root.crontab` file is edited, update the `cron` process with any changes to the `root.crontab` file.

Storing and Retrieving Data

The set of DataClass group parameters govern how FileServ processes the movement of data from disk to media. The directories controlled by DataClass group parameters are called migration directories because the data is moved (migrated) from disk to media. Movement of files from disk to media is also called file storage.

The following topics are discussed:

Store Files

Storing data begins by copying files into migration directories on disk. A file that remains in a migration directory is stored to media when the `fspolicy -s` command, for that DataClass group, is initiated by the system administrator or through the `cron` process.

Note

For more information on the `fsstore` command, refer to the *Command Reference* book.

In addition to providing an immediate store functionality, the `fsstore` command allows the user to override, on a file basis, the default values for the DataClass parameters that affect storage processing. Two parameters affect how the file data is handled during the storage processing. The first parameter specifies the cleanup action and the second specifies the number of secondary file copies.

Under usual circumstances, files in migration directories are routinely stored to media by initiation of the storage policy by cron. If data must be stored to media before the scheduled application of storage policy, the fsstore command can be issued for individual files.

When running the `fsstore` command, the user need not specify the media type used to store the data. FileServ selects the media. However, the user can override the default media type by specifying a different media type. An available media that has sufficient space and contains the same class of data is selected. If no appropriate nonblank media are available, the software uses media of the appropriate DataClass group (in which all files are deleted), or unused blank media.

The user specifies a minimum file size for storage, the drive pool to use for storage, and the number of file copies to create. The user also specifies that the file is not to be clustered.

Clean Up Data from Disk

Cleanup is the removal of data from disk after the application of the storage policy. Cleanup is specified with one of the commands below:

- `fsstore.`
- `fspolicy -t.`

> **Note**
>
> The `fspolicy` command cannot be used to simultaneously store and cleanup disk data.

- `fschfiat.` You can overrides the cleanup action for a specific file by changing the file's attributes through the `fschfiat` command. The changes made through this command always apply for the file, unless they are changed again with the `fschfiat` command.
- `fsrmcopy.`

Cleanup by File Name

Step 1. Run the `fsrmcopy` command. Specify the names of one or more files to remove from disk.

```
% fsrmcopy test1
```

Step 2. FileServ releases the associated data blocks.

Cleanup by DataClass

A DataClass group parameter specifies the default cleanup action. A Cleanup policy can be:

- Completely disabled.
  - Files can be excluded from clean-up. Use this only for frequently accessed files.
- Immediate truncation following the storage policy.
  - Valid only if all copies defined by the file's attributes are stored. If all copies are not made, the cleanup action does not occur, because that file is not eligible for cleanup.
  - The file remains on disk after storage policy until the `fspolicy -t` command is invoked for the file system on which the file resides, and the file is eligible for truncation.
- Truncated when cleanup policy determines that disk space is necessary.
  - The file remains on disk after storage policy until the `fspolicy -t` command is invoked for the file system on which the file resides, and the file is eligible for truncation.

Copy Secondary Files

For protection of media-based files, secondary file copying is provided. If enabled for a DataClass group, files automatically have a secondary copy maintained on a separate media.

A secondary file copy is written at the time the file is stored with the `fsstore` command by specifying the number of copies (including the primary file) up to the maximum number of copies set for the DataClass group. The number of secondary copies specified for a file cannot exceed the DataClass `maxcopies` parameter value.

Note

The current maximum allowable value for *maxcopies* is two copies, a primary and a secondary.

You can override the DataClass number of copies for a specific file with the `fschfiat` command.

General Storage Processing

When a storage request is received, either with the application of the storage policy or the `fsstore` command, the file eligibility is checked for storage.

- If no media copy exists or the file is changed since the last copy was made, the file is eligible for storage.
- If the file data is already on media and the file has not changed since the copy was made, the file is not eligible for storage to media.

Note

Before you can use the `fsstore` command, you must have read, write, or execute permission on the file you want to truncate.

After a file is eligible for storage, the file data for the primary copy is written to media. A volume is chosen for storage as either a default for the DataClass group (specified when the DataClass group is defined by the system administrator), or as a media type specified by the user when the `fsstore` command is issued.

If storage of a secondary copy is requested, the file data is copied to another media of the specified (or default) media type for the DataClass group associated with the file. Secondary copies are **not** placed on the same media as the primary copy.

If the DataClass group file cleanup parameter is set for immediate truncation, the file data is truncated after all copies have been stored.

Multiple storage requests queued for the same file are compared for compatibility. The first request is always processed. Subsequent storage requests are considered compatible when the specified media type is the same and the specified cleanup action is the same as the first request. When the requests are not compatible, the requests are failed with an indication that an incompatible store is already in progress.

Subsequent requests process the additional copies. All requests are retained until the storage process is complete. When the storage process is complete, all requests receive the same completion indication.

Retrieve Files

Files that reside only on media are retrieved to disk when accessed by the user during usual operations. Files can also be retrieved to disk by issuing a `fsretrieve` command. The `fsretrieve` command requires the name of each file to be retrieved. Multiple files can be retrieved with the same command.

Note

For more information on the `fsretrieve` command, refer to the *Command Reference* book.

However, if these files reside on different file systems, the command will either retrieve the files that are the most numerous, or the files that are listed first in the `fsretrieve` command. The command will not retrieve files from both Filesystem.

If a file is retrieved to a new file name, the new file name must be valid, and the user must have access to the directory. If the file exists, the user must also have access to the file. The group assigned to this new file is the primary group of the user who initiates the `fsretrieve` command.

For example, if the user belongs to the groups `FileServ`, `sysadm`, and `sw` respectively, all files retrieved to disk from media are in the `FileServ` group. The UNIX permissions for the new file are determined by the current `umask` of the user. If a failure occurs when retrieving a file to a new file name, the incomplete file with the new file name is not removed from disk. The user can delete the new file name or re-retrieve the file to the same new file name.

The `fsretrieve` command usually retrieves the primary copy of the file requested. If the primary copy is inaccessible or corrupted, the command defaults to the secondary copy. Completion status returns when the files are successfully copied onto disk. The `fsfileinfo` command can be used to determine if a file is on tape, disk, or both.

The file in the migration directory is associated with its media copy. When the file on disk is modified, but has not been stored to media again, the version on the media is invalidated and cannot be retrieved from the media with the `fsretrieve` command.

The file version that resides on disk is always assumed to be the current version. If the file is removed from disk using the UNIX `rm` command, the file is logically deleted from the media.

Note

When the UNIX `rm` command is used to remove a file, only the Primary copy of the file can be recovered with the fsundelete command. The `rm` command removes the Secondary copy of the file from the database.

The file data still resides on media, but it is no longer tracked and can only be retrieved from the trashcan using the `fsundelete` command. This retrieval is valid until `fsclean` is used to purge the trashcan.

Retrieve Files from Trashcan

FileServ recovers a particular version of a migrated file that was modified or removed from an archival directory on disk. As migrated files are modified or removed, an entry is placed in the trashcan with information to recover the file. Only the primary copy of the file can be recovered, not the secondary copy. When archival directories are removed from disk, an entry is also placed in the trashcan. The entries in the trashcan contain information necessary for the recovery of the file or directories.

The FileServ `fsundelete` command allows the user to recover files that were modified or removed from disk as listed below:

- Generate a Trashcan report.
- Restore deleted files or directories.
- Revert a file or directory.

The `fsundelete` command is used to determine what files are in the trashcan. The command is specified with either a full path name, filename, or partial filename. FileServ displays a list of files and directories owned by the user. These files match the requested file name. Optionally, the `fsundelete -p` command displays files and directories for which the user has UNIX read and write permissions.

Another report option restricts the report length, based on a specified start time and end time. Report entries are displayed for each user-accessible file within the specified time range.

Note

For more information on the `fsundelete` command, refer to the *Command Reference* book.

The `fsundelete -u` recovers a previous version of a migrated file or deleted migration directories. The command is specified with the full path name and restores the FileServ pointer to the area of the media where the data is stored.

This operation is successful if an entry for the file, or directory exists in the trashcan. The file is recovered on tape only as a migrated file. All files are restored to the directory and name under which they were stored when they were deleted. A restored file or directory can be moved to a different directory or renamed. If more than one version of a file exists in the trashcan, the file version to recover can be specified with the `-v` option. If the file already exists on disk, a force flag (`-f`) causes the recovered file to overwrite the existing disk file. For each file or directory requested, FileServ returns an error to the user if the file cannot be created and the `fsundelete` processing for that file is terminated.

If a file is moved within a directory (renamed with the `mv` command), the file name is not updated in the FileServ database. If the file is deleted, the previous name appears in the trashcan.

Note

If a file is moved within a directory, the file appears in the trashcan with the previous name.

Clean up Trashcan

Another command that must be routinely run is `fsclean`. The `fsclean` command purges the FileServ trashcan information.

Caution

Use of `fsclean` eliminates the ability to recover files from the trashcan with the `fsundelete` command!

Precautions must be taken when using the `fsclean` command. Trashcan serves as the repository for references to media data that is deleted from the disk, or modified. Each time a file is modified, its tape copy becomes invalid. An entry is created in the trashcan for the older version of the file. Also, when files (containing current file copies on media) are removed (UNIX `rm`), the trashcan receives an entry for each removed file. Because of asynchronous processing, an `fsclean` performed immediately after file removal may not completely clean the media. A `fsmedinfo -l` report is run to verify all files were deleted from the media. If files still exist, the `fsclean` command must be repeated.

The information can be purged by the *mediaID*, by *endtime* or both. The `-m` or `-t` option is required to remove information from the trashcan. To delete the entire trashcan, use the `fsclean` command with the `-t` option, and no time specified.

The `fsundelete` command allows recovery of removed or modified copies of files as long as an entry exists in the trashcan. When `fsclean` is run for media, all files on that media referenced in the trashcan are no longer recoverable with `fsundelete`.

Retrieve Partial Files

The `fsretrieve` command supports partial file copy to disk from media. A new file name is required, as well as a start and ending byte for the file data. With these parameters, the part of the file specified in the byte range is copied onto disk into the new file name.

The byte range is *zero relative*, meaning that if the file size (in bytes) is known, any part of the file is retrieved by specifying a range of bytes between 0 and the last byte (file size -1). File size information can be obtained with the UNIX `ls` command or the `fsmedinfo` report.

Note

For more information on the `fsretrieve` command, refer to the *Command Reference* book.

Step 1. Run the `fsretrieve` command. Specify one or more file names to retrieve. Or, specify a single file name to retrieve.

Step 2. Specify the `-n` *newfilename* into which you want to retrieve the specified file. The new filename must be in a local file system. Retrieval to an NFS-mounted file system is not permitted. Usually, the primary copy of the file is retrieved.

Note

Partial retrieval requires the use of the **-n** *newfilename* parameter.

Use other options as appropriate.

Retrieve Secondary Files

If a file is stored to media as both a primary and secondary copy, the secondary copy of the file can be retrieved to disk as a new file by using the `fsretrieve -c` command. This is the only command that allows access to the secondary copy of a file.

Delayed Dismount

FileServ provides the system administrator with the ability to configure each FileServ drive to delay the dismount of media after store and retrieve operations. The `fsconfig` command allows the system administrator to specify a delay time-out for each drive in the library subsystem.

Note

For more information on the `fsconfig` command, refer to the *Command Reference* book.

The delay time-out specifies the number of seconds since the last release before the media is dismounted. Media remains mounted in a drive until the specified time has passed since the last media release. The timer is cancelled when media is reallocated. The timer is reset to the time-out value when the media is released again.

Media is always dismounted if a currently delayed piece of media cannot fulfill an incoming request and all other drives are in use or delayed. Media marked suspect during the operation for which it is allocated are also dismounted.

Media delayed in a drive can be dismounted with the `fsdismount` command. The `fschstate` or `fsstate` commands can be used to determine if a drive is in a delayed-dismount status.

Note

For more information on these commands, refer to the fsaddclass command in the *Command Reference* book.

Media Management

This section describes operations required by both the VolServ operator and the FileServ system administrator for each media management function.

WARNING

BE CAREFUL when interacting with the mechanical interfaces to the automated libraries. Ensure hands, long hair, and loose clothing are clear of the entry port during normal media operations

DO NOT operate the robot when the library is open. (Use `fschstate -s off` before opening the library.)

The following topics are discussed:

Media Services

VolServ performs media services to manage the physical libraries and movement of media.

MediaClass Group Definitions

There are 54 MediaClass names with the following form:

`FileServID_mediatype_mediaclasstype`

There are ten possible values for FileServ ID: F0 through F9.

Six values of `mediaclasstype` exist: ADDBLANK or ADDBLNK, CHECKIN, IMPORT, DATA, MIGRATE, and REMOVE. An example of a valid MediaClass name is F0_D2S_ADDBLANK.

The VolServ operator must choose a MediaClass name for the media being entered. These MediaClass names are used exclusively for FileServ systems. In addition, the correct MediaClass name must be used for the media being entered. For example, Checked-out media being entered into the system is associated with `F#_###_CHECKIN`.

When entering media, the VolServ operator uses the IMPORT, ADDBLANK, and CHECKIN MediaClass names. Removal of media requires an automatic reclassification of MediaClass groups.The definitions of how each MediaClass name used by FileServ is shown in the table below:

Media Class Type	Definition
`REMOVE`	All media removed from FileServ with `fsmedout` is automatically reclassified from DATA to REMOVE.
`ADDBLANK`	Blank media being entered into the system.
`CHECKIN`	Checked-out media being re-entered into the system.

Media Class Type	Definition
DATA	All media entered into the system used by FileServ is automatically reclassified to this MediaClass group after the `fsmedin` command is performed successfully
MIGRATE	Media classified as DATA that are filled to a percentage equal to or exceeding FileServ system parameter `PERCENT_FULL_TO_MIGRATE` value is reclassified as MIGRATE and can be migrated to a different library mode based on the configuration of VolServ library action.

Load and Unload Media

Loading and removing media in the storage subsystems is a two-step process. Media are physically entered and removed from the system through the VolServ software. FileServ logically enters and removes media.

All media are physically entered into the storage subsystem through VolServ. This media are then logically entered into FileServ with the `fsmedin` command. Media entered into FileServ are normally formatted immediately. FileServ software allows media to be formatted immediately or withheld from format.

Media are logically removed from FileServ with the `fsmedout` command, then physically removed from the storage subsystem by the VolServ software.

Different methods of entering and removing media to and from the subsystems exist, depending on whether the media are blank or contain data (nonblank). Each of these methods is handled differently by FileServ.

Remove Media

Step 1. Run the `fsmedout` command with the media identifiers and the `-r` option for the media to be checked out. The `-t` *labeltext* option is not supported at this time.

Step 2. The physical location of the media (`-l` *location)* is used for tracking the media and is limited to 255 characters.

Media are logically checked out of the FileServ system.

Step 3. Contact the VolServ operator to complete the operation and to obtain the media.

Add Blank Media

FileServ uses the `fsmedin -b` command to logically enter blank media into FileServ software.

Note

For more information on the `fsmedin` command, refer to the *Command Reference* book.

Blank media can be added to the general blank pool or to a specific DataClass group pool. Blank media in the general blank pool are available for use in any DataClass group. Blank media in the DataClass group pool are only available for use by that particular DataClass group.

The quantity of media being added can be optionally entered as an argument of the `fsmedin -b` command. The quantity option is primarily used when a specific number, less than the value specified by the system `VS_DEF_QUANTITY` parameter, is needed.

Each type of media supported by FileServ has a default media length specified in the system parameters; `-DEF_MED_SPC_mediatype`. If two media of the same type have a different length, the `-l` option of the `fsmedin` command overrides the default value. For example, if two libraries are used the default media length can only apply to one library. Therefore, whenever adding blank media to the other library, the -l option must be used.

Step 1. Before new media is introduced to the storage subsystem, a barcode label must be applied to the outer edge of the media. If the label on media is marred when the media is entered, the media is automatically ejected from the storage subsystems. In this case, remove and replace the label. Blank media are entered by the VolServ operator using the Import command and the MediaClass `F#_XXX_ADDBLANK`.

Step 2. Run the `fsmedin` command with the `-b` option.

Step 3. Specify the `-q` *quantity* option to enter a specific quantity of media. The maximum number that can be specified is 99.

Step 4. Define the media type (`-t`) to be used. The media type is required for a system that handles multiple media types.

Remove Blank Media

The `fsmedout` command allows the removal of blank media.

Note

For more information on the `fsmedout` command, refer to the *Command Reference* book.

Two ways exist to remove blank media from the FileServ domain:

- Checkout blank media.
- Remove blank media.

Check Out Blank Media

Checking out blank media is NOT a common practice. Checking out blank media temporarily removes the media, but leaves the media information intact. In an extreme case, blank media can be removed from the system to free up media slots for other media.

The `fsmedout -r` command can temporarily remove blank media, retain information about the media in the FileServ database, and logically remove the media from the FileServ domain.

The status of Checked-out media is changed to `OUT OF FS`. Media can be physically removed from the storage subsystem by the VolServ operator. The VolServ operator is prompted and performs an Eject function to physically remove the media.

Checked-out blank media can be re-entered into the FileServ system using the `fsmedin -r` command. If the re-entry of the Checked-out media is no longer required, the `fsrminfo` command is used to remove all knowledge of the media from the FileServ system.

Caution

If you use the `fsrminfo` command to remove files, both the primary and secondary copies are removed. However, **only** the primary copy can be recovered.

Remove Blank Media

The `fsmedout -b` command allows removal of blank media. The media specified in the `fsmedout` command is logically removed from FileServ, then physically removed from the storage subsystem by the VolServ operator after performing an Eject function.

The issuer of the `fsmedout` request is notified that the operation is successful when the logical remove is complete. If media (specified for removal) is not found or does not fit the criteria specified in the options on the command, a message is returned for that media and the process continues for the remaining media specified.

After media is assigned to a DataClass group for file storage, it stays assigned to that DataClass group, even when all of the files on the media are deleted. The media reverts back to a blank status (depending on the cleanup action specified for the DataClass group). The `fsmedout` command removes blank media from the general blank pool or blanks allocated to a specific DataClass group.

Format Media

All blank media introduced into the storage subsystem can be submitted for immediate formatting or withheld from formatting. Media withheld (-w) from formatting is automatically formatted when chosen for data storage or by manually issuing the `fsformat` command.

Note

For more information on the `fsformat` command, refer to the *Command Reference* book.

If media is formatted, it is rejected when the drive attempts the format. The `-f` option for fsformat applies only to D-2 media. It overrides the rejection to reformat the media. Rejection of preformatted media without the force option prevents accidental formatting of media that contains data. Formatting D-2 media is time-consuming.

Media Duplication

Media duplication is intended for the following uses:

- Media maintenance - If errors frequently occur when attempting to read or write to the media.
- Data maintenance - If the media is filled with unusable space because of deleted files.
- Data duplication - A copy of the information is needed, in addition to the copy stored in the library.

Duplicating the contents of media or specific files on media is performed with:

- `fsmedcopy` command.
- `fsfilecopy` command.

Both types of duplications are described below.

Note

For more information on the `fsmedcopy` or `fsfilecopy` commands, refer to the *Command Reference* book.

Copy Media

The `fsmedcopy -r` command is used to replace media by moving data onto newer, better media or reclaiming the wasted space because of fragmentation.

Note

Running the `fsmedcopy -r` command is time consuming because of the update of file information. The extensive database activity caused by `fsmedcopy -r` affects the performance of most other FileServ commands. It is recommended to run `fsmedcopy -r` during a time of little or no use of FileServ.

Data on the specified media is copied onto other media (specified media, blank media, or a different media type not specified by the DataClass group). Multiple *copy to* media can be used if the destination media is not specified. Only active files on the original media are copied onto destination media. No trashcan contents are copied. After `fsclean` is invoked, any trashcan contents are removed and the original media are marked as blank. The original media can stay within the DataClass group or can be moved to the general blank pool, depending on the DataClass group definition. If `fsclean` is not performed, the original media remain within the DataClass group and new files are written. Any trashcan contents located on the media can be recovered using `fsmedinfo -l` and `fsundelete`. The `fsmedcopy` report can be used along with this option for finding candidate media for defragmentation.

Copy Files

The `fsfilecopy -r` command functions like the `fsmedcopy` command, except it replaces specified files, instead of the entire media.

A single file from media or several files from several media can be replaced. The only restriction is that all files must be in the same DataClass group. Specified files from the original media are moved onto either a blank media, specified media, or a different media type not defined by the DataClass group. Only active files can be copied. No trashcan contents are copied.

Media States

FileServ supports the following tape media:

- 3490, 3490E, and 3480.
- D-2.
- 8590.
- CTIII and CTIV.

Media can be placed in various states by the `fschmedstate` command.

The `fsmedinfo` report can be used to determine the present state of media, count of suspect errors, and how media is managed.

Note

For more information on the `fsmedinfo` command, refer to the *Command Reference* book.

The valid states for media in the FileServ system include the following:

State	Description
Protect	Reserved for read-only operations.
Unprotect	Available for read and write operations.
Available	Error free and resident in the storage subsystem.
Unavailable	Not used because it is removed from the storage subsystem or has extensive errors.
Suspect	A read/write or position error occurred with that media in a drive.
Unsuspect	Free from errors and problems.
Unmark	Media is error free and not designated for any copy or check-out operations.
`Mark`	An error has occurred in the handling of media, or there were read or write errors, or it is marked for copy or checkout operations. To cancel a *Mark Status* of *Error* or *Checkout* action, change media to `unmark`. These are the only states against which the `unmark` state can be applied.
Noerr	The `noerr` state resets the threshold count to 0 for the media. If any SER values are exceeded during a read or write request, the threshold count is incriminated. Cumulative threshold errors indicate that a problem may exist with the media or the drive. Examine the system logs to aid in determining the source of the error.

Storage Limit

Every FileServ installation has a storage limit capacity. If the storage limit capacity is exceeded, E-mail is sent to the system administrator and a message is logged in the system log file. All Storage requests fail until the system administrator deletes a sufficient number of files to place the system under the storage limit.

The `fsusedspace` command shows, in Gigabytes (GB), the total amount of stored primary copy data in the storage subsystem. Additional storage space can be created by defragmenting media, removing seldom-used media, nonblank media, or removing file information for Checked-out media that are no longer required also creates storage space.

The `fsmedcopy` command generates a report of fragmented media and duplicates the contents of fragmented media onto blank or nonblank media.

Note

For more information on the `fsusedspace` and `fsmedcopy` commands, refer to the *Command Reference* book.

The *fill* and *fragmentation* levels can be specified to locate those media with a high percentage of fragmentation. If no values are specified, all media that contain any percent of fragmentation is listed.

The `fsmedout` and `fsrminfo` commands can be used to remove file information for Checked-out media.

Caution

If you use the `fsrminfo` command to remove files, both the primary and secondary copies are removed. However, **only** the primary copy can be recovered.

After the amount of used storage is under the storage limit, FileServ must be cycled to re-enable Imports and Store requests.

The STORE_LIMIT_NOTICE system parameter located in $FS_HOME/*sysparm* directory, defines the range, in GB that remain before the storage capacity is exceeded.

Note

The fs_sysparm file is located in the directory *$FS_HOME/*sysparms, where $FS_HOME is the directory where FileServ is installed.

When this range is reached, FileServFileServ sends an E-mail message to the FS_OWNER_ID notifying the user that the storage limit capacity threshold is exceed.

Generate Media Reports

Many media are distributed across various subsystems. Therefore, the requirements for reporting on these media necessitate a Query function. The Query function allows display of media information by location, state, DataClass group, and movement.

Media Information

The fsmedinfo report (or **Media Information** screen) targets specific media and displays the vital statistics, along with a listing of the contents. Media can be listed in categories of the current operational states and DataClass association by using the fsmedlist report. Each report shows a different picture of the current status of the media in the system.

Note

For more information on the fsmedlist and fsmedinfo commands, refer to the *Com*

Step 1. Run the `fsmedinfo` command and specify one or more media identifiers to report.

Step 2. Use the `-l` option to obtain a long report. An example of the output is shown below:

```
% fsmedinfo -l feu002

-----------------------------------------------------------------
 Media Information Report Tue Jan 26 17:32:44 1999
 Media ID: feu002(1)  (1 = primary, 2 = secondary)
 Media Type: D2 Small
-----------------------------------------------------------------

Storage Area: MAN_AREA1
     Storage Area:  VolServ
         Class ID:  document     Bytes Used:     15,000,000
    Last Accessed:  03-feb-1999 16:31:44 Space Remaining:25,277,000,000
     Media Status:  AVAIL      Percent Used:     0.06
    Write Protect:  N         Suspect Count:    0
      Mark Status:  UNMARKED    Mount Count:    3
         Dir File:  VALID   Threshold Count:    0
  Medium Location:  SLOT/BIN   Manufacturer:
         Formatted  Y              Batch ID:
  Number of Files:  4
External Location:  N/A

File system </arch1> is unmounted.
File system </arch3> is unmounted.
All monitored filesystems must be mounted to show “-l” output.

FS0000 26 68309 fsmedinfo completed: Command Successful.
```

Step 3. If the parent of an individual file cannot be found, the report shows question marks for the path and indicates that the parent is unknown. An example of this output is shown below. If this situation occurs, perform an audit (`fsaudit`) on the file system.

```
File Size Status Modify/Delete Date File Pathname
```

```
------------------------------------------------------------------
 3773 Active Mon Dec 7 16:58:11 1998 ???/xxx : Parent_Unknown
 3773 Active Mon Dec 7 16:58:13 1998 ???/yyy : Parent_Unknown
 3773 Active Mon Dec 7 16:58:16 1998 ???/zzz : Parent_Unknown
```

Media List

The `fsmedlist` command or `Media List` screen produces a list of all the media, or the media in a particular DataClass group. If the class is not specified, the report lists information for media in all DataClass groups and media in the general blank pool.

> **Note**
>
> Because a single piece of media can be in more than one category, the sum of media in all categories is not likely to match the total number of media in the class.

Step 1. Run the `fsmedlist` command and specify the appropriate options.

Step 2. An example of the short report is shown below:

```
% fsmedlist -c testclass

------------------------------------------------------------------
Media List Report                Mon Jan 18 10:46:42 1999
Class ID
Drive   Slot  Trans Exit  Out  Blank   Prot  Avail   Susp  Mark  Total
------------------------------------------------------------------

testclass
       0     2     0     0    0      1     0      1     1     2002
FS0000 18 02946 fsmedlist completed: Command Successful.
```

Step 3. The `-l` option generates a longer report shown below that includes the identifiers of the media for each of the categories previously described. If no options are specified, all of the media identifiers in each category are listed, as shown below. The list is sent to *`stdout`* and can be redirected to a file or piped to a printer.

```
Media List Report    Mon Jan 18 11:43:27 1999
Class ID:  vrpclass
-----------------------------------------------------------------

In Drive
        Total:0
In Bin
                Media IDs:   jbd4(0)   vrp002(0)   vrp003(0)
                             vrp005(1)    vrp006(0)
        Total:5
In Transit
        Total:0
Exiting
        Total:0
Out of adic
        Total:0
Marked for Check-out
        Total:0
Marked for Copy
        Total:0
Mark Error
        Total:0

Marked
        Total:0
Formatted Blank
                Media IDs:   jbd4(0)
        Total:1

S0000 28 41651 fsmedlist completed: Command Successful.
```

NOTES

3

Utilities

Roadmap

Topic	Refer To Chapter
Initialize FileServ, configure interface with VolServ, perform start up and shut down, modify system configuration, and tune system for better performance.	1
Manage: DataClass, files, migration, data, and media.	2
Using the FileServ utilities.	3
Troubleshoot operating problems.	4

Utilities

The `$FS_HOME/util` directory contains software tools used to simplify various operations. An explanation of each software tool is explained in the `.README` files.

fspic

The `fspic` utility displays a status of the FileServ system including:

- Hardware components.
- Media mounted in drives.
- Number of minutes since the media was accessed.

This utility also monitors disk usage for all migration file systems under FileServ, as well as certain database locations.

All information is updated at regular intervals.

autostart scripts

The `autostart` scripts are platform-specific. The scripts were added to the `$FS_HOME/util/autostart` directory. These scripts allow the user to automatically:

- Start and stop the `fs_dmapi` daemon.
- Mount and unmount the FileServ controlled file systems.
- Start and stop the `lock_manager` process.
- Start up and stop FileServ.

Starting the `fs_dmapi` and `lock_manager` processes, and mounting of the FileServ controlled DMAPI file system must be done prior to starting FileServ. These functions can be performed manually, but the `autostart` scripts allow this to occur automatically. For more information, refer to the `.README` file on these scripts.

FileServ and Amass Autostart

If FileServ and AMASS are on the same platform, the FileServ processes must be started before the AMASS processes. If the autostart scripts are enabled for both FileServ and AMASS, the FileServ autostart script must be executed before the AMASS autostart script.

Assign a name to the FileServ autostart script so that it will start before the AMASS autostart script. An example of the FileServ and AMASS autostart script names int /etc/rx2.d directory is shown below:

```
lrwxrwxr-x 1 root sys 25 Jan 16 08:58 S95FileServ ->\
/etc/init.d/auto_FileServ

lrwxrwxr-x 1 root sys 25 Jan 16 08:58 S99amass ->\
/etc/init.d/amass
```

dbcheck

The `dbcheck` utility checks the consistency of a database by validating the location and key values associated with each record and key in the data and key files.

Note

Run the `dbcheck` utility only when FileServ is not running.

ADIC recommends that you run this utility on a regular basis. Database inconsistencies will be reported with a message describing the nature of the error and the database address of the record involved.

This utility is located in the `$FS_HOME/util` directory. Options are described in the table below:

Option	Description
-s	Enables the checking of set consistency.

Option	Description
-k	Performs key file structure checking.
-dk	Causes `dbcheck` to ensure the key file contains the key for each key field in each data file.
-kd	Causes `dbcheck` to ensure that a record exists for each key in each key file.
-ts	Performs timestamp consistency checks.
-r#	Reports percentage complete to the file `stderr.`
-p#	Sets to the # the number of pages in the cache for use in `dbcheck.`
-f#	Sets to # the number of open files for `dbcheck.`
-t	Prints a traceback of the B-tree when a key file inconsistency is detected.
-c	Prints a count of objects scanned in the check.
dbname	To determine the name of the database, look in the `$FS_HOME/internal/fsdb` directory for files ending with the `.dbd` extension.

If you experience unusual behavior when running a FileServ command, look under `$FS_HOME/internal/fsdb` for any data or key files that are unusually large. These files will have a `.d01` or `.k0x` extension, where "x" is a number between 1 and 5. Unusually large file size may indicate there is a problem with the database. If running `dbcheck` indicates a key file error, run `keybuild` to correct the problem.

keybuild

The `keybuild` utility will rebuild all key files for a database. It rebuilds the key files by first reinitializing the file, and then sequentially reading all records from each data file and recreating each key field from the contents of the record.

Step 1. From the `$FS_HOME/util` directory enter the following:

```
# keybuild dbname
```

where:

Option	Description
dbname	Name of the database whose key files you want to rebuild.

Step 2. This utility recreates the key files when there is a key file inconsistency, as reported by `dbcheck`.

4

Trouble-shooting Tools

Roadmap

Topic	Refer To Chapter
Initialize FileServ, configure interface with VolServ, perform start up and shut down, modify system configuration, and tune system for better performance.	1
Managing: DataClass, files, migration, data, and media.	2
Using the FileServ utilities.	3
Troubleshoot operating problems.	4

Data Retrieval Fails

If a data retrieval operation fails, the returned status message usually provides information to determine the cause of the failure.

Reissuing the request usually corrects the problem.

However, if a message appears indicating one of the reasons below is responsible for the failure, additional troubleshooting is needed before reissuing the request.

- A database access error.
- An abnormal process failure (a process abnormally terminates or is killed).
- A communication message is not received.
- Resources are not allocated (no resources available).
- The FileServ software terminates (either abnormally or intentionally).
- A file or media access problem.

Commands that ID Problems

The commands in the table below will help you to further troubleshoot problems. For information on these commands, refer to the `man` pages.

Commands	Commands
`brc(1M)`	`master(4)`
`bru(1)`	`mt(1)`
`close(2)`	`mtio(7)`
`cpio(1)`	`open(2)`
`dd(1M)`	`reboot(1M)`
`dump(1M)`	`restore(1M)`
`hinv(1M)`	`system(4)`
`ioctl(2)`	`tar(1)`
`lboot(1M)`	`tps(7M)`
`ls(1)`	`uname(1)`
`MAKEDEV(1M)`	`vi(1)`

Disaster Recovery

The `fsdump`, `fsrestore`, and `fsaudit` commands provide disaster recovery in case of a FileServ disk crash. For procedures for using these commands, refer to the *Command Reference* book.

Messages

Messages are displayed in response to a command or when certain events occur. Messages are divided into two categories:

- Status messages.
- Syslog messages.

Every status and syslog message that results from a FileServ command includes a request identifier. The request identifier is a unique number assigned to each request at the point it is initiated in the FileServ system.

Status Messages

Status messages are returned to the user in response to a every command issued by that user. They are displayed on the command line where the command is invoked.

Status messages are classified into one of the following categories.

Category	Definition
Usage	User entered a command with invalid syntax. This message contains the proper syntax for the command and applies only to the command line interface.
Interim	Give interim status as the command is processed.
Completion	Signify the completion of the command and indicate if the command completed successfully or unsuccessfully.

Syslog Messages

Syslog messages are sent to a logging facility to inform a system administrator of important system conditions, to record additional information about these conditions, or to record the occurrence of an event. Each message has an associated priority that determines how the logging facility processes the message.

Console messages are viewed from a shell window on the FileServ system console.

Because certain unsolicited messages are not passed to FileServ for display at the FileServ system console, ADIC recommends that the host `syslog` monitor be displayed from the FileServ system console.

Extract Syslog Messages

The `fsextlog` command extracts information from any of the standard UNIX syslog files.

Note

For more information on the `fsextlog` command, refer to the *Command Reference* book.

The information is returned to `stdout`. All information in the specified log is extracted by default. The time range (`-t`) option extracts information stored in the log only over the specified time period.

Log File Maintenance

The script `newfslog` retains copies of syslogs messages. The script increments each message by adding an extension to each message; **.0, *.1*, … Seven messages are retained by default; however, the script can be modified to retain more messages based on the sites need. The script is executed by the `cron` process.

The `log_monitor` script is a alternative to `newfslog` for log maintenance. This script can be run as needed to refresh log files, or called from `cron` to monitor the logs and prevent them from filling up the `$FS_HOME` file system. Log cleaning will only be triggered when the command line parameters are met.

The system variables `$FS_HOME, $LOG_PATH,` and `$NONARCH_LOG_PATH` must be defined before the script will run.

`$LOGPATH` is the path to the archival directory where old log files (files that should not be stored in an archival directory) will be stored. Type `log_monitor` without parameters to obtain information on the parameter options. If `log_monitor` returns "Command not found" make sure that the path on line one is correct for the Perl processor, and change as necessary. Information on past `log_monitor` activity can be found in `$NONARCH_LOG_PATH/log_monitor.log.`

Set Up FileServ Logging

Step 1. Open a FileServ environment shell window.

Step 2. Look at the /etc/syslog.conf file to make sure that the console window on the workstation is receiving FileServ syslog messages. This file contains entries directing the pertinent FileServ syslog messages to the FileServ system host computer.

Step 3. Log in as root and enter the following command:

```
# -ps -elf | grep syslogd
```

Step 4. Enter the following command to display the daemon pid of the syslogd:

```
# kill -HUP daemonpid
```

Step 5. Issue the UNIX `tail` command of the file `logfile`.

```
% tail -f $FS_HOME/syslogs/logfile
```

As host messages are generated, they are displayed in this window.

Step 6. Determine the log file where you can extract information, for example, `$FS_HOME/syslogs/logfile`.

Step 7. Change directory (`cd`) to the directory where the log files are stored and make sure that they exist.

Step 8. Run the `fsextlog` command specifying the log file where information is extracted.

```
% fsextlog logfile1 -t 1999:01:06 1999:01:07
```

Use the `-t` option to extract only those messages that were logged between *starttime* and *endtime.* The *starttime* and *endtime* must be before the current time and the times must be specified as `YYYY:MM:DD:hh:mm:ss`.

Extracted logs go to `stdout` but may be redirected.

Recover Data

To support data recovery due to a disk crash, the following tasks are required:

Task	Page
Dump Data	4-9
Restore Data	4-11
Audit Database	4-17

Dump Data

The `fsdump` command is used to perform a dump of each file system.

Note

For more information on the `fsdump` command, refer to the *Command Reference* book.

Enter a `-f` *dumpfilename* to specify the file system to be dumped.

The `fsdump` command creates a `fs_dump.file` for each file system dumped. The file system must be mounted for the file to be created. The file is dumped with the files in the file system. Then, the file is removed from disk.

When used with the UNIX `xfsrestore` command and the FileServ `fsaudit` command, any file system that contains migration directories can be restored.

The `fsdump` command also supports recovery for backup of nonmigration file systems. A log message advises the system administrator that the file system is not archival. The dump continues, but the `fs_dump.file` does not exist and is not dumped to the dump tape. The `fsdump` command does not continue if the file system does not exist, another dump procedure is in progress, or for improper syntax.

Checkpoint the file systems should be done on a regular basis to allow for data recovery from a disaster.

Step 1. To ensure the integrity of the dump file before checkpointing, run the UNIX `fsck` utility for the file system.

Step 2. Verify the file system name by viewing the `/etc/fstab` file.

The */arch3* file system is used throughout this example.

Step 3. Run `fsaudit -f` on the file system to be dumped to cleans up any discrepancies on the file system.

Step 4. Because the trashcan data will not be valid on a file system recovered from a dump file, run `fsclean -t`.

Tip
ADIC recommends that you always run `fsclean -t` prior to running `fsdump`.

Step 5. To verify the file system's integrity before dumping it, run the ULNIX `xfs_check` utility.

```
% xfs_check /dev/dsk/dks2d6s7
```

If problems are discovered, correct them before continuing.

Step 6. Mount a tape on a drive.

Step 7. Run the `fsdump` command with the arguments appropriate to the platform as specified in the man page. Enter the destination file or drive and any optional arguments if appropriate.

```
% fsdump -f /dev/rmt/tps0d3 /arch
```

The `fsdump` utility prompts the user to change tapes if more than one tape is required. A sequence number is given each tape, starting at 1.

- Change tapes when prompted.
- Label tapes with the sequence number so they can be restored in the proper sequence.

Restore Data

File systems that are checkpointed with the `fsdump` command are restored with the `fsxrestore` command.

Note

Restored files and directories have the same permissions as when they were dumped.

If new relation points have been added, run `fsdump` so these directories are located on the dump tape. The `fsaudit` command fails if all parents of relation points are not on the dump tape.

The `fs_dump.file` is also restored to disk for use by the `fsaudit -r` command.

Restoration of the entire file system is not required if a dump tape is lost or bad. However, restoration is recommended.

Step 1. Verify the new system is large enough to contain all the data restored from tape by using the UNIX `df` command.

The */arch3* file system is used throughout this example.

```
% df
FILESYSTEM        TYPE  BLOCKS  USE     AVAIL  USE  MOOUNTED ON
/dev/dsk/dks2d6s7 xfs   2081008 1524392 556616 74   arch3
```

Step 2. Run the `xfsrestore` command.

```
% xfsrestore -f /dev/rmt/tps0d3 /arch3
```

Step 3. Checkpoint the FileServ database. For information, see “Dump Data” on page 4-9.

Step 4. Perform a recovery audit by using `fsaudit -r -f`. For information, see “Audit Database” on page 4-17.

If the `fsaudit` command fails with a message that indicates a relation parent unknown, the relation point parents are not on the dump tape. This situation occurs if:

- All files are not recovered from the dump tape (bad dump tape).
- Dump is not made after a DataClass relation is added to a directory.

Use the UNIX `mkdir` command to create the parent and rerun the `fsaudit -r -f` command.

If the procedure fails, call technical support.

Step 5. Run the `keyBuildAll` utility to update the key files in the database.

Step 6. Run `fsaudit -f` twice on the recovered file system.

- The first time `fsaudit -f` will clean up discrepancies left by the audit recovery.
- The second time it should not find any problems. If it does find a problem, continue running `fsaudit -f` until all remaining discrepancies are cleared.

Step 7. To remove invalid candidate files, perform the following:

```
% cd $FS_HOME/internal/policy_dir/
     store_candidates
rm *
% cd ../trunc_candidates
% rm *
```

Step 8. To rebuild the candidate lists, run the following:

```
% fspolicy -s -y <all archival\
               filesystems> -w

% fspolicy -t -y <all archival\
              filesystems> -w
```

Restore Data in Interactive Mode

Step 1. Verify that the disk file system is large enough to contain all the data restored from tape by using the UNIX `df` command.

```
% df
FILESYSTEM        TYPE  BLOCKS  USE     AVAIL  USE  MOOUNTED ON
/dev/dsk/dks2d6s7 xfs   2081008 1524392 556616 74   arch
```

Step 2. Run the `xfsrestore` -i command.

```
% xfsrestore -f /dev/rmt/tps0d3
/arch3
```

An interactive shell is invoked with the -> prompt.

Step 3. Run the UNIX `ls` command at the `xfsrestore` prompt to verify which directories and files are in the dumped file system.

```
% -> ls
```

Step 4. Add the appropriate relation point parents from the dumped file system to the extraction list to be restored.

```
-> add home
-> add mrktg
-> add junk
```

Note

If you specify a directory, it and all of its descendants are added to the extraction list, unless the h option is used on the command line.

All relation points in the dumped file system are specified on the extraction list. All relation point parents must exist.

If a failure occurs, add the missing relation points to the extraction list and continue.

Step 5. Add any additional files and directories from the dump tape to the extraction list. File names are entered on a single line or one at a time.

List the files and directories and verify that all files to be extracted from tape are starred (*). A star by a file name indicates it is selected for extraction.

Step 6. Enter the `extract` command.

```
-> extract
```

The selected directories and files are restored to disk. If more than one tape contains the dumped data, the user is prompted when to mount the next volume.

Type the sequence number of the tapes used for the `fsdump`.

After all selected directories and files are restored, type `q` or `quit` to exit the `xfsrestore` interactive mode.

Step 7. Checkpoint the FileServ database. For information, see "Dump Data" on page 4-9.

Step 8. Perform an audit of the file system using the `fsaudit -r -f` command. For information, see "Audit Database" on page 4-17.

If the `fsaudit` command fails with a message that indicates a relation parent unknown, the relation point parents are not on the dump tape. This situation occurs if:

- All files are not recovered from the dump tape (bad dump tape).
- Dump is not made after a DataClass relation is added to a directory.

Use the UNIX `mkdir` command to create the parent and rerun the `fsaudit -r -f` command.

If the procedure fails, call technical support.

Step 9. Run the `keyBuildAll` utility to update the key files in the database.

Step 10. Run `fsaudit -f` twice on the recovered file system.

- The first time `fsaudit -f` will clean up discrepancies left by the audit recovery.
- The second time it should not find any problems. If it does find a problem, continue running `fsaudit -f` until all remaining discrepancies are cleared.

Step 11. To remove invalid candidate files, perform the following:

```
% cd $FS_HOME/internal/policy_dir/
     store_candidates
rm *
% cd ../trunc_candidates
% rm *
```

Step 12. To rebuild the candidate lists, run the following:

```
% fspolicy -s -y <all archival filesystems> -w
% fspolicy -t -y <all archival filesystems> -w
```

Audit Database

The `fsaudit` command serves two purposes:

- Recovery of file systems.
- Database maintenance compares UNIX file system information and FileServ database information. The `fsaudit` command for database maintenance does not use the `fs_dump.file`.

> **Note**
>
> For more information on the `fsaudit` command, refer to the *Command Reference* book.

File System Recovery

In a recovery situation, `fsaudit` uses the `fs_dump.file` to change old file handles of the `fsdump` files to the new file handles of the restore or `xfsrestore` files. After running `fsaudit`, run `fsdump` to create a new dump tape for future restoration activities.

> **Tip**
>
> ADIC recommends that you audit a file system once a week.

If a disk entry is not found for a file, the disk entry is created during the reconciliation portion. For missing directories, the UNIX `mkdir` command is used. For missing files, the file is created as a truncated file.

Database Maintenance

The FileServ database contains UNIX file system information to track the following:

- Migration file systems.
- Media where files are stored.
- Association points in the UNIX file system.
- Files in directories that are related to DataClass groups.

Running fsaudit Command

FileServ and database software must be online and running for execution of fsaudit. The file system that is specified for fsaudit cannot be active when the command is invoked or it fails.

The `fsaudit` command disallows users, with the exception of `root`, from accessing the file system during an audit.

Step 1. Checkpoint the FileServ database.

Step 2. Confirm that FileServ is active.

Step 3. Run the `fsaudit` command and specify the file system to audit. The name must be entered exactly as it appears in the `/etc/fstab` file.

Use the appropriate options as needed.

Step 4. If the file system mount point name mount point changes from the name that the `fsdump` command executed, specify the `-o` *oldfilesystem* name.

Step 5. If the file system is restored to a lower point in the directory hierarchy, specify `-d`, the name of the directory in which the file system is restored.

When using the `fsaudit -r` command, all parent directories of relation points must be present in the `fs_dump.file`. If a parent directory does not exist for a relation point in the `fs_dump.file`, the `fsaudit` command fails.

Troubleshoot System Performance

If the DataClass `mintime` parameter for the DataClass groups and `cron` setting that govern the frequency of the migration policy application for a DataClass group are set to too short a time, FileServ may be invoked too frequently. This is known as thrashing because the policy is applied so frequently that it prevents other system operations that have lower priority.

If thrashing occurs with a number of DataClass groups, it can cause a major impact on system performance. If system performance is degraded, the `syslog` file can be monitored for messages that indicate when the policy is applied.

Step 1. If `syslog` indicates that a low block call-out was issued, use the UNIX `df -a` (disk free space) command to monitor the disk fill level of all of the file systems.

Step 2. Run the `fspolicy` command.

Use the appropriate options as needed. For command information, refer to the *Command Reference* book.

Step 3. Run the `fsmodclass -m` command with the DataClass group identifier to shorten the time limit for the *mintime* parameter for DataClass groups in the file system where this parameter is set relatively high.

Reports

The table below lists the reports generated by FileServ:

DataClass Report

To view the parameter settings for each DataClass identifier, follow the steps below:

Step 1. Run the `fsclassinfo` command. Use the names of a specified DataClass group on which to report.

Step 2. An example of the output for a long report is displayed below:

```
% fsclassinfo test1 -l

-------------------------------------------------------------------
 Class Information Report           Tue Dec 29 09:11:27 1998
 Class ID: test1
-------------------------------------------------------------------

   Soft Limit:      80 Max Copies:             2
   Hard Limit:      95 Media Type:             D2S
   Notify ID:       root File Spanning:        n
   Security Code:   XXDD File Cleanup:         MINTIME
```

```
    Account Number:  12345 Media Cleanup:       CLASS
    Drive Pool:      fspool File Clustering:    n
    Default Copies:  1 Mintime (min):           10

Associated Directories:
 /data/joe/prod001
 /data/june/prod002

FS0000 31 174105 fsclassinfo completed: Command Successful.
```

Media Fragmentation Report

To view a list of media suspected as fragmented, follow the steps below:

Step 1. Run the `fsmedcopy` command. The level of fragmentation is defined by options submitted with the command

Use other appropriate options as needed. For command information, refer to the *Command Reference* book.

Step 2. The output displayed below list all media that are at least 25 percent used. Of that 25 percent, at least 25 percent is wasted or unusable space.

```
% fsmedcopy -f 25 -w 25

-----------------------------------------------------------------------
Media Fragmentation Report                  Tue Dec 15 13:32:39 1998
-----------------------------------------------------------------------

Media ID    Fill Level              Wasted Space
ht0e01      55.11                   28.20
222003      53.74                   76.12
75d004      76.48                   44.90
000005      88.00                   92.46
343456      85.92                   75.87

FS0000 15 00895 fsmedcopy completed: Command Successful.
```

In the example for media 000005, the amount of wasted space is 92.46 percent, which represents. 81.36 percent of the total space.

Media Movement Report

To view a report showing media that has been removed or added to the storage system, follow the steps below:

Step 1. Run the `fsmoverpt` command to generate a report that lists the movement of all media.

Use other appropriate options as needed. For command information, refer to the *Command Reference* book.

Step 2. An example of the output is displayed below:

```
% fsmoverpt

------------------------------------------------------------------------
 Media Movement Report                        Tue Dec 15 13:42:12 1998
------------------------------------------------------------------------

MediaID    MediaType    Date/Time           Movement     Comments
srr001     D2 small     Dec 4 08:52:12      reentered
cah005     D2 small     Dec 15 13:15:59     added
jdd006     D2 small     Dec 15 13:16:09     added
999007     D2 small     Dec 15 13:16:18     added
986008     D2 small     Dec 15 13:16:29     added
765009     D2 small     Dec 15 13:16:42     added
```

Resource Queue Report

To view a snapshot of FileServ requests waiting for resources, follow the steps below:

Step 1. To view all resource requests associated with your storage system, run the `fsqueue` command without options as shown below:

```
% fsqueue

-------------------------------------------------------------------
Resource Queue State                          Fri Aug 29 13:35:57 1999

Request Volserv    Request Priority State    Drive Media  Submitted
ID      ID         Type    C:M:A             ID    ID
13720   424187015  STR     5:5:20   PROCESS  N/A   N/A    Aug 29 1999
13720   424187015  STR     5:5:20   PROCESS  N/A   N/A    Aug 29 1999
-------------------------------------------------------------------

FS0000 03 13786 fsqueue completed: Command Successful.
```

The table below lists the Request Types and State fields:

Request	Description		States	Description
ENT	Enter Media		Queue	Request in queue
EJE	Eject Media		Ready	Ready to process
CPY	Copy		Format	Formatting
FMT	Format		Verify	Verify label
INS	Inspect		Mount	Mounting
N/A	Not Available		Process	Being Processed
RTV	Retrieve		Copy	Start copying
STR	Store		Cancel	Being cancelled
UNK	Unknown or Not Available		Complete	Request completed

Step 2. Use other appropriate options as needed. For command information, refer to the *Command Reference* book.

History Report

To view a history of media and components, follow the steps below:

Step 1. Run the `fshistrpt` command with the `-m` *mediaID* option for a media history report. A example of the output is shown below:

```
% fshistrpt -m cah001
-------------------------------------------------------------------

Media History                                Tue Dec 15 13:59:44 1998
Report

Media ID          Media Type                 Date/Time State Change
Comments

-------------------------------------------------------------------

cah001            D2 Small                   Dec 4 09:24:31protect
USER=root

cah001            D2 Small                   Dec 4 12:94:27protect
USER=root
```

Step 2. Or, run the `fshistrpt` command with the `-h` option for a component history report. An example of the output is shown below:

```
% fshistrpt -h V0, 10, V0, 11, V0, 12, V0, 13, V0, 203
-------------------------------------------------------------------
Component History Report   Tue AUG 24 10:42:48 1998
-------------------------------------------------------------------

Component ID  Status Change        Date/Time        Media In Use
V0,203        OFFLINE             AUG 3            10:40:02
V0,12         ONLINE              AUG 3            10:50:40
V0,10         ONLINE              AUG 3            10:54:30
V0,10         ONLINE              AUG 3            10:54:30
V0,30         ONLINE              AUG 3            10:04:20
V0,10         IN USE              AUG 3
```

Component Statistics Report

To view a snapshot of current device states in the storage systems, follow the steps below:

Step 1. Run the `fschstate` command without any options to generate a report that shows all storage subsystems and the state of FileServ.

Step 2. Use other appropriate options as needed. For command information, refer to the *Command Reference* book.

An example of the output is shown below:

```
% fschstate
------------------------------------------------------------------------

Component Status                                Thu Jan 21 17:27:52 1999
Report

Component Alias    Drive ID     State        Status       Media ID

AMLJ_CTIII_DRO     101          ON           FREE         NONE

AMLJ_CTIII_DR1     102          ON           FREE         NONE

VolSub             N/A          ON           N/A          N/A

The FileServ system is active.
FS0000 07     1464244 fschstate completed: Command Successful.
```

* Drives not configured in FileServ.

+ Media mounted in a drive is unknown to FileServ.

Hardware Configuration Report

To view a report of current drives and subsystems configured into the storage systems, follow the steps below:

Step 1. Run the `fsconfig` command without options to generate a report for all components.

> **Note**
>
> If the argument contains a space, put single quotes around the argument.

Step 2. Use other appropriate options as needed. For command information, refer to the *Command Reference* book.

An example of the output is shown below:

```
% fsconfig
------------------------------------------------------------------------

Hardware Configuration Report       Thur Feb 4 16:41:41 1999
Component ID: V0,10

Device pathname:                    /dev/er90/s0
user Alias:                         ER90_DR1
Component Type:                     DRIVE
Drive Type:                         ER90
Drive ID:                           10
Delay Time:                         0

Hardware Configuration Report       Thur Feb 4 16:41:41 1999
Component ID: V0, 11

Device pathname:                    /dev/er90/s1
User Alias:                         ER90_DR2
Component Type:                     DRIVE
Drive Type:                         ER90
Drive ID:                           11
Delay Time:                         0

Hardware Configuration Report       Thur Feb 4 16:41:41 1999
Component ID: V0, 12

Device pathname:                    /dev/er90/s2
User Alias:                         ER90_DR3
Component Type:                     DRIVE
Drive Type:                         ER90
Drive ID:                           12
Delay Time:                         0

FS0000 22 02843 fsconfig completed: Command Successful.
```

Contact Technical Support

Before contacting technical support, follow the steps below:

Step 1. Obtain the following information:

- FileServ Serial Number: _____
- Host ID: _____

Step 2. Contact technical support and be prepared to supply the above information to them.

Technical support can be reached as follows:

- In the USA and Canada, call 1-800-827-3822.
- Outside the USA and Canada, call 303-874-0188 or toll-free 00800-9999-3822.
- techsup@adic.com

NOTES

Index

D

E

F

G

I

L

R

S

T

U

V

X